IBM COBOL for Linux en x86 1.2

Guía de programación

SC28-3118-01

Nota

Antes de utilizar esta información y el producto al que da soporte, asegúrese de leer la información general en “Avisos” en la página 665.

Segunda edición

Esta edición se aplica a la versión 1.2 de IBM® COBOL for Linux® en x86 (número de programa 5737-L11) y a todos los releases y modificaciones posteriores hasta que se indique lo contrario en nuevas ediciones. Asegúrese de que está utilizando la edición correcta para el nivel del producto.

Puede ver o descargar publicaciones en copia software de forma gratuita en COBOL for Linux en la biblioteca x86.

Contenido

Tablas

Prefacio

Acerca de esta información

Bienvenido a IBM COBOL for Linux on x86, IBMpara Linux on x86.

Esta información describe el uso del compilador IBM COBOL y entorno de ejecución para Linux en x86, al que se hace referencia en esta información como COBOL para Linux.

Existen algunas diferencias entre el host y la estación de trabajo COBOL. Para obtener detalles sobre las diferencias de lenguaje y sistema entre COBOL para Linux y Enterprise COBOL para z/OS, consulte "Migración de Enterprise COBOL for z/OS a COBOL for Linux en x86" en la *Guía de migración*.

Cómo le ayudará esta información

Esta información le ayudará a escribir, compilar, enlazary ejecutar programas IBM COBOL for Linux en x86 .

Esta información presupone experiencia en el desarrollo de programas de aplicación y algunos conocimientos de COBOL. Se centra en utilizar COBOL para cumplir los objetivos de programación y no en la definición del lenguaje COBOL. Para obtener información completa sobre la sintaxis COBOL, consulte la publicación *COBOL for Linux en x86 Consulta de lenguaje*.

Esta información también presupone que está familiarizado con Linux. Para obtener información sobre Linux, consulte la documentación del sistema operativo.

Términos abreviados

Algunos términos se utilizan de forma abreviada en esta información. Las abreviaturas de los nombres de productos utilizados con más frecuencia se listan alfabéticamente en la tabla siguiente.

Término utilizado	**Formato largo**
TXSeries	IBM TXSeries for Multiplatforms
CICS TX	CICS TX Advanced o CICS TX Standard
CICS	IBM TXSeries for Multiplatforms o IBM CICS TX
COBOL para Linux	IBM COBOL for Linux en x86
Db2	IBM Db2 for Linux, UNIX y Windows

Además de estos términos abreviados, el término "85 COBOL Estándar" se utiliza en esta información para hacer referencia a la combinación de los estándares siguientes:

- ISO 1989:1985, lenguajes de programación-COBOL
- ISO/IEC 1989/AMD1:1992, lenguajes de programación-COBOL: módulo de función intrínseca
- ISO/IEC 1989/AMD2:1994, Lenguajes de programación-Corrección y aclaración para COBOL
- ANSI INCITS 23-1985, Idiomas de programación-COBOL
- ANSI INCITS 23a-1989, Lenguajes de programación-Módulo de función intrínseca para COBOL
- ANSI INCITS 23b-1993, Lenguaje de programación-Enmienda de corrección para COBOL

Otros términos, si no se entienden comúnmente, se muestran en *cursiva* la primera vez que aparecen y se listan en el glosario.

Cómo leer diagramas de sintaxis

Utilice la descripción siguiente para leer los diagramas de sintaxis de esta información:

- Lea los diagramas de sintaxis de izquierda a derecha, de arriba abajo, siguiendo la ruta de la línea.

  El símbolo ▶▶— indica el principio de un diagrama de sintaxis.

  El símbolo —▶ indica que el diagrama de sintaxis continúa en la línea siguiente.

  El símbolo ▶— indica que el diagrama de sintaxis continúa desde la línea anterior.

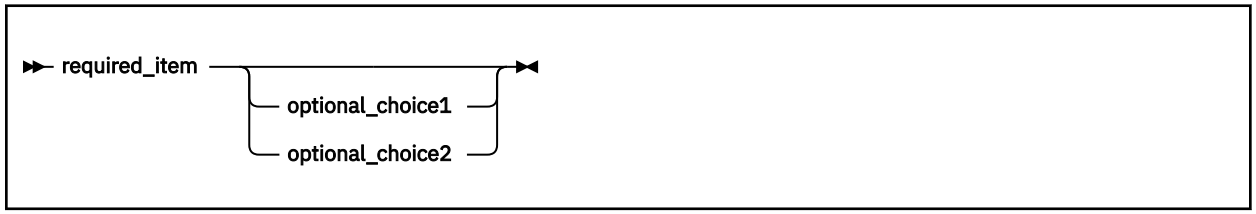
  El símbolo —▶◀ indica el final de un diagrama de sintaxis.

  Los diagramas de unidades sintácticas que no sean sentencias completas empiezan con el símbolo ▶— y terminan con el símbolo —▶ .
- Los elementos necesarios aparecen en la línea horizontal (la ruta principal).

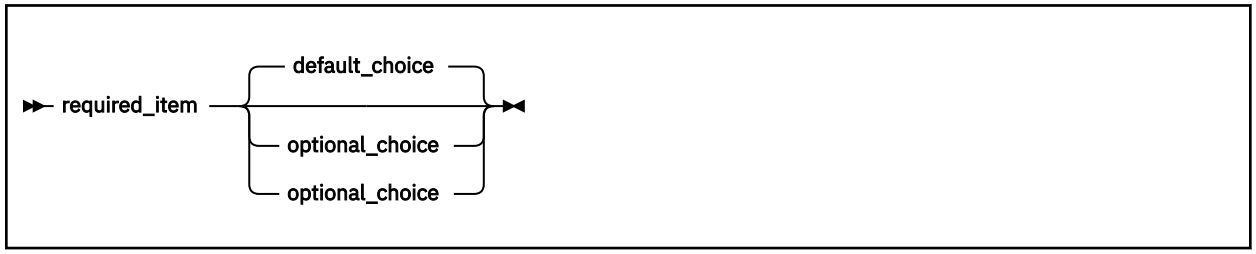
- Los elementos opcionales aparecen debajo de la ruta principal.

- Si puede elegir entre dos o más elementos, aparecen verticalmente, en una pila. Si debe elegir uno de los elementos, un elemento de la pila aparece en la vía de acceso principal.

  Si elegir uno de los elementos es opcional, toda la pila aparece debajo de la ruta principal.



  Si uno de los elementos es el valor predeterminado, aparece encima de la ruta principal y las opciones restantes se muestran a continuación.



- Una flecha que vuelve a la izquierda, por encima de la línea principal, indica un elemento que se puede repetir.

Si la flecha de repetición contiene una coma, debe separar los elementos repetidos con una coma.

- Las palabras clave aparecen en mayúsculas (por ejemplo, FROM). Se deben deletrear exactamente como se muestra. Las variables aparecen en minúsculas (por ejemplo, *nombre-columna*). Representan valores o nombres proporcionados por el usuario.
- Si se muestran signos de puntuación, paréntesis, operadores aritméticos u otros símbolos de este tipo, debe especificarlos como parte de la sintaxis.

Cómo utilizar ejemplos

Esta información muestra numerosos ejemplos de sentencias COBOL de ejemplo, fragmentos de programa y programas pequeños para ilustrar las técnicas de codificación que se están discutiendo. Los ejemplos de código de programa se escriben en minúsculas, mayúsculas o mayúsculas y minúsculas para demostrar que puede escribir sus programas de cualquiera de estas maneras.

Para separar más claramente algunos ejemplos del texto explicativo, se presentan en a `monospace font`.

Las palabras clave COBOL y las opciones de compilador que aparecen en el texto se muestran generalmente en SMALL UPPERCASE. Otros términos como nombres de variables de programa se muestran a veces en *una fuente en cursiva* para mayor claridad.

Si copia y pega ejemplos de la documentación en formato PDF, asegúrese de que los espacios de los ejemplos (si los hay) están en su lugar; es posible que tenga que añadir manualmente algunos espacios que faltan para asegurarse de que el texto de origen COBOL se alinea con las columnas necesarias según la sección "Formato de referencia COBOL" en la *Referencia de lenguaje*. De forma alternativa, puede copiar y pegar ejemplos de la documentación de formato HTML y los espacios ya deben estar en su lugar.

Información relacionada

La información de esta Guía de programación está disponible en línea en la documentación de IBM COBOL for Linux en https://www.ibm.com/docs/en/cobol-linux-x86/1.2. El sitio web IBM Documentation también tiene *COBOL for Linux en x86 Consulta de lenguaje*.

Cómo enviar sus comentarios

Sus comentarios son importantes para ayudarnos a proporcionar información precisa y de alta calidad. Si tiene comentarios sobre esta información o cualquier otra documentación de COBOL para Linux , envíe sus comentarios a: compinfo@cn.ibm.com.

Asegúrese de incluir el nombre del documento, el número de publicación, la versión de COBOL para Linuxy, si procede, la ubicación específica (por ejemplo, el número de página o la cabecera de sección) del texto sobre el que está comentando.

Cuando envía información a IBM, concede a IBM un derecho no exclusivo para utilizar o distribuir la información de cualquier forma que IBM considere adecuada sin incurrir por ello en ninguna obligación con el usuario.

Accesibilidad

Las características de accesibilidad ayudan a los usuarios que tienen una discapacidad, como por ejemplo movilidad restringida o visión limitada, a utilizar correctamente los productos de tecnología de la información.

Funciones de accesibilidad

IBM COBOL for Linux en x86 utiliza el estándar W3C más reciente, WAI-ARIA 1.0, para garantizar la conformidad con US Section 508 y Web Content Accessibility Guidelines (WCAG) 2.0. Para aprovechar las características de accesibilidad, utilice el release más reciente del lector de pantalla en combinación con el navegador web más reciente soportado por este producto.

Navegación mediante teclado

Este producto utiliza teclas de navegación estándar.

Información de interfaz

Puede utilizar el software de reconocimiento de voz como una herramienta de texto a voz (TTS) para ver la salida generada por el producto.

La documentación del producto en línea está disponible en la documentación de IBM , que se puede ver desde un navegador web estándar.

Los archivos PDF tienen un soporte de accesibilidad limitado. Con la documentación en PDF, puede utilizar la ampliación de fuentes opcional, los valores de visualización de alto contraste y puede navegar solo con el teclado.

Para que el lector de pantalla pueda leer con precisión los diagramas de sintaxis, los ejemplos de código fuente y el texto que contiene los símbolos PICTURE de punto o coma, debe establecer el lector de pantalla para que hable todos los signos de puntuación.

Información de accesibilidad relacionada

Además de los sitios web de soporte y servicio de asistencia técnica estándar de IBM , IBM ha establecido un servicio telefónico TTY para que los clientes sordos o con dificultades auditivas puedan acceder a los servicios de soporte y ventas:

Servicio de teletipo 800-IBM-3383 (800-426-3383) (en Norteamérica)

IBM y accesibilidad

Para obtener más información sobre el compromiso que IBM tiene con la accesibilidad, consulte IBM Accesibilidad.

Parte 1. Codificación del programa

Capítulo 1. Estructuración del programa

Acerca de esta tarea

Los programas COBOL constan de cuatro divisiones: `IDENTIFICATION DIVISION`, `ENVIRONMENT DIVISION`, `DATA DIVISION`y `PROCEDURE DIVISION`. Cada división tiene una función lógica específica.

Para definir un programa, sólo es necesario el `IDENTIFICATION DIVISION` .

Tareas relacionadas
“Identificación de un programa” en la página 3
“Descripción del entorno informático” en la página 5
“Descripción de los datos” en la página 9
“Proceso de los datos” en la página 14

Identificación de un programa

Utilice `IDENTIFICATION DIVISION` para asignar un nombre a un programa y, opcionalmente, proporcionar otra información de identificación.

Acerca de esta tarea

Puede utilizar los párrafos `AUTHOR`, `INSTALLATION`, `DATE-WRITTEN`y `DATE-COMPILED` opcionales para obtener información descriptiva sobre un programa. Los datos que especifique en el párrafo `DATE-COMPILED` se sustituyen por la fecha de compilación más reciente.

```
IDENTIFICATION DIVISION.
Program-ID.    Helloprog.
Author.        A. Programmer.
Installation.  Computing Laboratories.
Date-Written.  30/06/2020.
Date-Compiled. 05/07/2020.
```

Utilice el párrafo `PROGRAM-ID` para nombrar el programa. El nombre de programa que asigne se utiliza de estas maneras:

- Otros programas utilizan ese nombre para llamar a su programa.
- El nombre aparece en la cabecera de cada página, excepto la primera, del listado de programas que se genera al compilar el programa.

Consejo: Si un nombre de programa distingue entre mayúsculas y minúsculas, evite las discrepancias con el nombre que busca el compilador. Verifique que el valor adecuado de la opción de compilador `PGMNAME` está en vigor.

Tareas relacionadas
“Cambio de la cabecera de un listado de origen” en la página 4
“Identificación de un programa como recursivo” en la página 4
“Cómo marcar un programa como invocable conteniendo programas” en la página 4
“Establecimiento de un programa en un estado inicial” en la página 4

Referencias relacionadas

Convenios para nombres de programa (*COBOL for Linux en x86 Consulta de lenguaje*)

Identificación de un programa como recursivo

Codifique el atributo RECURSIVE en la cláusula PROGRAM-ID para especificar que un programa se puede volver a especificar de forma recursiva mientras una invocación anterior todavía está activa.

Acerca de esta tarea

Sólo puede codificar RECURSIVE en el programa más externo de una unidad de compilación. Ni los subprogramas anidados ni los programas que contienen subprogramas anidados pueden ser recursivos.

Atención: Una sentencia PERFORM no debe hacer que se ejecute a sí misma. Esto constituye un PERFORM recursivo, que puede provocar resultados imprevisibles. Por lo tanto, no debe especificar sentencias PERFORM recursivas.

Consulte "Ejemplo: secciones de almacenamiento" en la página 12, que muestra que un programa recursivo utiliza WORKING-STORAGE y LOCAL-STORAGE. Tenga en cuenta que un programa recursivo sólo tendrá 1 copia de WORKING-STORAGE, pero una nueva copia de LOCAL-STORAGE para cada CALL.

Tareas relacionadas
"Compartir datos en programas recursivos" en la página 14
"Realización de llamadas recursivas" en la página 470

Cómo marcar un programa como invocable conteniendo programas

Utilice el atributo COMMON en el párrafo PROGRAM-ID para especificar que un programa puede ser llamado por el programa contenedor o por cualquier programa del programa contenedor. El programa COMMON no puede ser llamado por ningún programa contenido en sí mismo.

Acerca de esta tarea

Sólo los programas contenidos pueden tener el atributo COMMON .

Conceptos relacionados
"Programas anidados" en la página 459

Establecimiento de un programa en un estado inicial

Utilice la cláusula INITIAL en el párrafo PROGRAM-ID para especificar que siempre que se llame a un programa, dicho programa y cualquier programa anidado que contenga se colocarán en su estado inicial.

Acerca de esta tarea

Cuando un programa se establece en su estado inicial:

- Los elementos de datos que tienen cláusulas VALUE se establecen en los valores especificados.
- Las sentencias GO TO y las sentencias PERFORM cambiadas se encuentran en sus estados iniciales.
- Los archivos noEXTERNAL se cierran.

Tareas relacionadas
"Finalización y reintroducción de programas o subprogramas principales" en la página 457

Referencias relacionadas
"WCLEAR" en la página 305

Cambio de la cabecera de un listado de origen

La cabecera de la primera página de un listado fuente contiene la identificación del compilador y el nivel de release actual, la fecha y hora de compilación y el número de página.

Acerca de esta tarea

El ejemplo siguiente muestra estos cinco elementos:

```
PP 5737-L11 IBM COBOL for Linux 1.2.0     Date 04/21/2023   Time 17:38:17   Page     1
```

La cabecera indica la plataforma de compilación. Puede personalizar la cabecera en las páginas siguientes del listado utilizando la sentencia TITLE de dirección del compilador.

Referencias relacionadas
sentencia TITLE (*COBOL for Linux en x86 Consulta de lenguaje*)

Descripción del entorno informático

En el ENVIRONMENT DIVISION de un programa, se describen los aspectos del programa que dependen del entorno informático.

Acerca de esta tarea

Utilice CONFIGURATION SECTION para especificar los elementos siguientes:

- Sistema para compilar el programa (en el SOURCE-COMPUTER párrafo)
- Sistema para ejecutar el programa (en el OBJECT-COMPUTER párrafo)
- Elementos especiales como el signo de moneda y los caracteres simbólicos (en el SPECIAL-NAMES párrafo)
- Clases definidas por el usuario (en el REPOSITORY párrafo)

Utilice los párrafos FILE-CONTROL y I-O-CONTROL de INPUT-OUTPUT SECTION para:

- Identifique y describa las características de los archivos del programa.
- Asocie los archivos con el nombre de archivo del sistema correspondiente, directa o indirectamente.
- Opcionalmente, identifique el sistema de archivos (por ejemplo, SFS o STL) que está asociado con un archivo. También puede hacerlo en tiempo de ejecución.
- Proporcione información sobre cómo se accede a los archivos.

"Ejemplo: párrafo FILE-CONTROL" en la página 5

Tareas relacionadas
"Especificación del orden de clasificación" en la página 6
"Definición de caracteres simbólicos" en la página 7
"Definición de una clase definida por el usuario" en la página 8
"Identificación de archivos en el sistema operativo (ASSIGN)" en la página 8

Referencias relacionadas
Secciones y párrafos (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: párrafo FILE-CONTROL

El ejemplo siguiente muestra cómo el párrafo FILE-CONTROL asocia cada archivo del programa COBOL con un archivo físico conocido en el sistema de archivos. Este ejemplo muestra un párrafo FILE-CONTROL para un archivo indexado.

```
SELECT COMMUTER-FILE (1)
  ASSIGN TO COMMUTER (2)
  ORGANIZATION IS INDEXED (3)
  ACCESS IS RANDOM (4)
  RECORD KEY IS COMMUTER-KEY (5)
  FILE STATUS IS (5)
```

```
    COMMUTER-FILE-STATUS
    COMMUTER-STL-STATUS.
```

(1)
La cláusula SELECT asocia un archivo en el programa COBOL con un archivo del sistema correspondiente.

(2)
La cláusula ASSIGN asocia el nombre del archivo en el programa con el nombre del archivo tal como lo conoce el sistema. COMMUTER puede ser el nombre de archivo del sistema o el nombre de la variable de entorno cuyo valor de tiempo de ejecución se utiliza como nombre de archivo del sistema con nombres de directorio y vía de acceso opcionales.

(3)
La cláusula ORGANIZATION describe la organización del archivo. Si omite esta cláusula, se presupone ORGANIZATION IS SEQUENTIAL .

(4)
La cláusula ACCESS MODE define la forma en que los registros del archivo están disponibles para su proceso: secuencial, aleatorio o dinámico. Si omite esta cláusula, se presupone ACCESS IS SEQUENTIAL .

(5)
Es posible que tenga sentencias adicionales en el párrafo FILE-CONTROL en función del tipo de archivo y del sistema de archivos que utilice.

Tareas relacionadas
"Descripción del entorno informático" en la página 5

Especificación del orden de clasificación

Puede utilizar la cláusula PROGRAM COLLATING SEQUENCE y la cláusula ALPHABET del párrafo SPECIAL-NAMES para establecer el orden de clasificación que se utiliza en varias operaciones en elementos alfanuméricos.

Acerca de esta tarea

Estas cláusulas especifican el orden de clasificación para las siguientes operaciones en elementos alfanuméricos:

- Comparaciones especificadas explícitamente en condiciones de relación y condiciones de nombre-condición
- Valores de HIGH-VALUE y LOW-VALUE
- SEARCH ALL
- SORT y MERGE a menos que se altere temporalmente mediante una frase COLLATING SEQUENCE en la sentencia SORT o MERGE

"Ejemplo: especificar la secuencia de clasificación" en la página 7

La secuencia que utilice se puede basar en uno de estos alfabetos:

- EBCDIC: hace referencia al orden de clasificación asociado con el juego de caracteres EBCDIC
- NATIVE: hace referencia al orden de clasificación especificado por el valor de entorno local. El valor de entorno local hace referencia al nombre de entorno local de idioma nacional en vigor en el momento de la compilación. Normalmente se establece durante la instalación.
- STANDARD-1: hace referencia al orden de clasificación asociado con el juego de caracteres ASCII definido por *ANSI INCITS X3.4, Juego de caracteres codificados-Código estándar nacional americano de 7 bits para intercambio de información (ASCII de 7 bits)*
- STANDARD-2: hace referencia al orden de clasificación asociado con el juego de caracteres definido por *ISO/IEC 646 -- Information technology -- ISO 7-bit coded character set for information interchange, International Reference Version*

- Una modificación de la secuencia ASCII que se define en el párrafo SPECIAL-NAMES

También puede especificar un orden de clasificación que defina.

Restricción: Si la página de códigos es DBCS, Extended UNIX Code (EUC) o UTF-8, no puede utilizar la cláusula ALPHABET .

La cláusula PROGRAM COLLATING SEQUENCE no afecta a las comparaciones que implican operandos nacionales o DBCS .

Tareas relacionadas
“Elección de secuencias de clasificación alternativas” en la página 170
“Comparación de datos nacionales (UTF-16)” en la página 205
Capítulo 11, “Establecimiento del entorno local”, en la página 213
“Control de la secuencia de clasificación con un entorno local” en la página 219

Ejemplo: especificar la secuencia de clasificación

El ejemplo siguiente muestra la codificación ENVIRONMENT DIVISION que puede utilizar para especificar un orden de clasificación en el que las letras mayúsculas y minúsculas se manejan de forma similar en las comparaciones y en la ordenación y fusión.

Cuando cambia la secuencia ASCII en el párrafo SPECIAL-NAMES , la secuencia de clasificación general se ve afectada, no sólo la secuencia de clasificación de los caracteres que se incluyen en el párrafo SPECIAL-NAMES .

```
IDENTIFICATION DIVISION.
. . .
ENVIRONMENT DIVISION.
  CONFIGURATION SECTION.
   Object-Computer.
      Program Collating Sequence Special-Sequence.
   Special-Names.
      Alphabet Special-Sequence Is
          "A" Also "a"
          "B" Also "b"
          "C" Also "c"
          "D" Also "d"
          "E" Also "e"
          "F" Also "f"
          "G" Also "g"
          "H" Also "h"
          "I" Also "i"
          "J" Also "j"
          "K" Also "k"
          "L" Also "l"
          "M" Also "m"
          "N" Also "n"
          "O" Also "o"
          "P" Also "p"
          "Q" Also "q"
          "R" Also "r"
          "S" Also "s"
          "T" Also "t"
          "U" Also "u"
          "V" Also "v"
          "W" Also "w"
          "X" Also "x"
          "Y" Also "y"
          "Z" Also "z".
```

Tareas relacionadas
“Especificación del orden de clasificación” en la página 6

Definición de caracteres simbólicos

Utilice la cláusula SYMBOLIC CHARACTERS para asignar nombres simbólicos a cualquier carácter del alfabeto especificado. Utilice la posición ordinal para identificar el carácter, donde la posición 1 corresponde al carácter X'00 '.

Acerca de esta tarea

Por ejemplo, para dar un nombre al carácter más (X'2B' en el alfabeto ASCII), codifique:

```
SYMBOLIC CHARACTERS PLUS IS 44
```

No puede utilizar la cláusula SYMBOLIC CHARACTERS cuando la página de códigos indicada por el entorno local es una página de códigos de varios bytes.

Tareas relacionadas
Capítulo 11, “Establecimiento del entorno local”, en la página 213

Definición de una clase definida por el usuario

Utilice la cláusula CLASS para asignar un nombre a un conjunto de caracteres que se listan en la cláusula.

Acerca de esta tarea

Por ejemplo, asigne un nombre al conjunto de dígitos codificando la cláusula siguiente:

```
CLASS DIGIT IS "0" THROUGH "9"
```

Puede hacer referencia al nombre de clase sólo en una condición de clase. (Esta clase definida por el usuario no es la misma que una clase orientada a objetos.)

No puede utilizar la cláusula CLASS cuando la página de códigos indicada por el entorno local es una página de códigos de varios bytes.

Identificación de archivos en el sistema operativo (ASSIGN)

La cláusula ASSIGN asocia el nombre de un archivo tal como se conoce en un programa al archivo asociado que utilizará el sistema operativo.

Acerca de esta tarea

Puede utilizar una variable de entorno, un nombre de archivo del sistema, un literal o un nombre de datos en la cláusula ASSIGN . Si especifica una variable de entorno como nombre de asignación, la variable de entorno se evalúa en tiempo de ejecución y el valor (incluidos los nombres de directorio y vía de acceso opcionales) se utiliza como nombre de archivo del sistema.

Si utiliza un sistema de archivos que no sea el predeterminado, debe indicar el sistema de archivos explícitamente, por ejemplo, especificando el identificador del sistema de archivos antes del nombre de archivo del sistema. Por ejemplo, si MYFILE es un archivo STL y utiliza F1 como nombre del archivo en el programa, puede codificar la cláusula ASSIGN como se indica a continuación:

```
SELECT F1 ASSIGN TO STL-MYFILE
```

Si MYFILE no es una variable de entorno, o es una variable de entorno establecida en la serie vacía, el código mostrado anteriormente trata MYFILE como un nombre de archivo del sistema. Si MYFILE es una variable de entorno que tiene un valor en tiempo de ejecución que no es la serie vacía, se utiliza el valor de la variable de entorno.

Por ejemplo, si MYFILE se establece mediante el mandato `export MYFILE=RSD-YOURFILE`, el nombre de archivo del sistema es YOURFILEy el archivo se trata como un archivo RSD , alteración temporal del ID del sistema de archivos (STL) codificado en la cláusula ASSIGN .

Si escribe un nombre de asignación entre comillas o comillas simples (por ejemplo, `"STL-MYFILE"`), el valor de cualquier variable de entorno se ignora. Se utiliza el nombre de asignación literal.

Tareas relacionadas
“Variación del archivo de entrada o salida en tiempo de ejecución” en la página 9
“Identificación de archivos” en la página 117

Referencias relacionadas
“Prioridad de la determinación del sistema de archivos” en la página 121
“ARCHIVOS” en la página 317
Cláusula ASSIGN (*COBOL for Linux en x86 Consulta de lenguaje*)

Variación del archivo de entrada o salida en tiempo de ejecución

El *nombre-archivo* que codifica en una cláusula SELECT se utiliza como constante en todo el programa COBOL, pero puede asociar el nombre de dicho archivo con un archivo de sistema diferente en tiempo de ejecución.

Acerca de esta tarea

Cambiar un nombre de archivo dentro de un programa COBOL requeriría cambiar las sentencias de entrada y las sentencias de salida y volver a compilar el programa. De forma alternativa, puede cambiar el mandato *nombre-asignación* en `export` para utilizar un archivo diferente en tiempo de ejecución.

Los valores de variable de entorno que están en vigor en el momento de la sentencia OPEN se utilizan para asociar nombres de archivo COBOL a los nombres de archivo del sistema (incluidas las especificaciones de unidad).

Ejemplo: utilización de archivos de entrada diferentes
Este ejemplo muestra que puede utilizar el mismo programa COBOL para acceder a archivos diferentes estableciendo una variable de entorno antes de que se ejecuten los programas.

Considere un programa COBOL que contenga la siguiente cláusula SELECT :

```
SELECT MAINFILE ASSIGN TO MAINA
```

Supongamos que desea que el programa acceda al archivo `checking` o `savings` utilizando el archivo llamado `MAINFILE` dentro del programa. Para ello, establezca la variable de entorno MAINA antes de que se ejecute el programa utilizando una de las dos sentencias siguientes, según corresponda, Suponiendo que los archivos `checking` y `savings` están en el directorio `/accounts` :

```
export MAINA=/accounts/checking
export MAINA=/accounts/savings
```

Por lo tanto, puede utilizar el mismo programa para acceder al archivo `checking` o `savings` como archivo llamado `MAINFILE` dentro del programa sin tener que cambiar o volver a compilar el fuente.

Descripción de los datos

Defina las características de los datos y agrupe las definiciones de datos en una o más de las secciones de `DATA DIVISION`.

Acerca de esta tarea

Puede utilizar estas secciones para definir los siguientes tipos de datos:

- Datos utilizados en operaciones de entrada-salida: `FILE SECTION`
- Datos desarrollados para el procesamiento interno:
  - Para que el almacenamiento se asigne estáticamente y exista durante la vida de la *unidad de ejecución*: `WORKING-STORAGE SECTION`

  - Para que el almacenamiento se asigne cada vez que se entre un programa y se desasigne a la devolución del programa: LOCAL-STORAGE SECTION
- Datos de otro programa: LINKAGE SECTION

Conceptos relacionados
"Comparación de WORKING-STORAGE y LOCAL-STORAGE" en la página 11

Tareas relacionadas
"Utilización de datos en operaciones de entrada y salida" en la página 10
"Utilización de datos de otro programa" en la página 13

Utilización de datos en operaciones de entrada y salida

Defina los datos que utiliza en las operaciones de entrada y salida en FILE SECTION.

Acerca de esta tarea

Proporcione la siguiente información sobre los datos:

- Asigne un nombre a los archivos de entrada y salida que utilizará el programa. Utilice la entrada FD para dar nombres a los archivos a los que pueden hacer referencia las sentencias de entrada-salida en PROCEDURE DIVISION .

  Los elementos de datos definidos en FILE SECTION no están disponibles para las sentencias PROCEDURE DIVISION hasta que el archivo se haya abierto correctamente.
- En la descripción de registro que sigue a la entrada FD , describa los registros del archivo y sus campos. El nombre de registro es el objeto de las sentencias WRITE y REWRITE .

Los programas de la misma unidad de ejecución pueden hacer referencia a los mismos nombres de archivo COBOL.

Puede utilizar la cláusula EXTERNAL para programas compilados por separado. Cualquier programa de la unidad de ejecución que describa el archivo puede hacer referencia a un archivo definido como EXTERNAL .

Puede utilizar la cláusula GLOBAL para programas en una estructura anidada o contenida. Si un programa contiene otro programa (directa o indirectamente), ambos programas pueden acceder a un archivo común haciendo referencia a un nombre de archivo GLOBAL .

Puede compartir archivos físicos sin utilizar definiciones de archivos externas o globales en programas fuente COBOL. Por ejemplo, puede especificar que una aplicación tenga dos nombres de archivo COBOL, pero estos archivos COBOL están asociados con un archivo del sistema:

```
SELECT F1 ASSIGN TO MYFILE.
SELECT F2 ASSIGN TO MYFILE.
```

Conceptos relacionados
"Programas anidados" en la página 459

Tareas relacionadas
"Compartir archivos entre programas (archivos externos)" en la página 483

Referencias relacionadas
"Entradas de FILE SECTION" en la página 10

Entradas de FILE SECTION

Las entradas que puede utilizar en FILE SECTION se resumen en la tabla siguiente.

Tabla 1. ***Entradas de FILE SECTION***	
Cláusula	**Para definir**
FD	El *nombre-archivo* al que se hace referencia en las sentencias de entrada-salida de PROCEDURE DIVISION (OPEN, CLOSE, READ, STARTy DELETE). Debe coincidir con *file-name* en la cláusula SELECT . *nombre-archivo* se asocia con el archivo del sistema a través del *nombre-asignación*.
RECORD CONTAINS *n*	Tamaño de registros lógicos (longitud fija). El tamaño entero indica el número de bytes de un registro independientemente del USAGE de los elementos de datos del registro.
RECORD IS VARYING	Tamaño de los registros lógicos (longitud variable). Si se especifican tamaños o tamaños enteros, indican el número de bytes de un registro independientemente del USAGE de los elementos de datos del registro.
RECORD CONTAINS *n* TO *m*	Tamaño de los registros lógicos (longitud variable). Los tamaños enteros indican el número de bytes de un registro independientemente del USAGE de los elementos de datos del registro.
VALUE OF	Un elemento en los registros de etiqueta asociados con el archivo. Sólo comentarios.
DATA RECORDS	Nombres de registros asociados al archivo. Sólo comentarios.
RECORDING MODE	Tipo de registro para archivos secuenciales

Referencias relacionadas
FILE SECTION (*COBOL for Linux en x86 Consulta de lenguaje*)

Comparación de WORKING-STORAGE y LOCAL-STORAGE

El modo en que se asignan e inicializan los elementos de datos varía en función de si los elementos están en WORKING-STORAGE SECTION o LOCAL-STORAGE SECTION.

Cuando se invoca un programa, se asigna el WORKING-STORAGE asociado con el programa.

Los elementos de datos que tienen cláusulas VALUE se inicializan con el valor adecuado en ese momento. Durante la duración de la unidad de ejecución, los elementos WORKING-STORAGE persisten en su último estado utilizado. Las excepciones son:

- Un programa con INITIAL especificado en el párrafo PROGRAM-ID

  En este caso, los elementos de datos WORKING-STORAGE se reinicializan cada vez que se especifica el programa.
- Un subprograma que se llama dinámicamente y luego se cancela

  En este caso, los elementos de datos WORKING-STORAGE se reinicializan en la primera reentrada al programa después de CANCEL.

WORKING-STORAGE se desasigna al finalizar la unidad de ejecución.

Se asigna una copia separada de los datos de LOCAL-STORAGE para cada llamada de un programa, y se libera al devolver del programa. Si especifica una cláusula VALUE para un elemento LOCAL-STORAGE , el elemento se inicializa en ese valor en cada llamada. Si no se especifica una cláusula VALUE , el valor inicial del elemento no está definido.

"Ejemplo: secciones de almacenamiento" en la página 12

Tareas relacionadas
“Finalización y reintroducción de programas o subprogramas principales” en la página 457

Referencias relacionadas
WORKING-STORAGE SECTION (*COBOL for Linux en x86 Consulta de lenguaje*)
LOCAL-STORAGE SECTION (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: secciones de almacenamiento

El ejemplo siguiente es un programa recursivo que utiliza WORKING-STORAGE y LOCAL-STORAGE.

```
CBL apost,pgmn(lu)
*********************************
* Recursive Program - Factorials
*********************************
 IDENTIFICATION DIVISION.
 Program-Id. factorial recursive.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  numb  pic 9(4)  value 5.
 01  fact  pic 9(8)  value 0.
 LOCAL-STORAGE SECTION.
 01  num  pic 9(4).
 PROCEDURE DIVISION.
     move numb to num.

     if numb = 0
        move 1 to fact
     else
        subtract 1 from numb
        call 'factorial'
        multiply num by fact
     end-if.

     display num '! = ' fact.
     goback.
 End Program factorial.
```

El programa genera la salida siguiente:

```
0000! = 00000001
0001! = 00000001
0002! = 00000002
0003! = 00000006
0004! = 00000024
0005! = 00000120
```

Las tablas siguientes muestran los valores cambiantes de los elementos de datos en LOCAL-STORAGE y WORKING-STORAGE en las llamadas recursivas sucesivas del programa y en los gobacks subsiguientes. Durante los gobacks, `fact` acumula progresivamente el valor de 5! (cinco factoriales).

Llamadas recursivas	**Valor para num en LOCAL-STORAGE**	**Valor para numb en WORKING-STORAGE**	**Valor para fact en WORKING-STORAGE**
principal	5	5	0
1	4	4	0
2	3	3	0
3	2	2	0
4	1	1	0
5	0	0	0

Gobacks	Valor para num en LOCAL-STORAGE	Valor para numb en WORKING-STORAGE	Valor para fact en WORKING-STORAGE
5	0	0	1
4	1	0	1
3	2	0	2
2	3	0	6
1	4	0	24
principal	5	0	120

Conceptos relacionados
“Comparación de WORKING-STORAGE y LOCAL-STORAGE” en la página 11

Utilización de datos de otro programa

La forma en que se comparten los datos depende del tipo de programa. Los datos se comparten de forma diferente en los programas que se compilan por separado que en los programas anidados o en los programas que son recursivos o multihebra.

Acerca de esta tarea

Tareas relacionadas
“Compartir datos en programas compilados por separado” en la página 13
“Compartición de datos en programas anidados” en la página 13
“Compartir datos en programas recursivos” en la página 14
“Pasar datos” en la página 473

Compartir datos en programas compilados por separado

Muchas aplicaciones constan de programas compilados por separado que se llaman y pasan datos entre sí. Utilice `LINKAGE SECTION` en el programa llamado para describir los datos pasados desde otro programa.

Acerca de esta tarea

En el programa de llamada, codifique una sentencia `CALL . . . USING` para pasar los datos.

Tareas relacionadas
“Pasar datos” en la página 473
“Codificación de la SECCIÓN DE ENLACE” en la página 476

Compartición de datos en programas anidados

Algunas aplicaciones constan de programas anidados, es decir, programas contenidos en otros programas. Los elementos de datos Level-01 pueden incluir el atributo `GLOBAL` . Este atributo permite que cualquier programa anidado que incluya las declaraciones acceda a estos elementos de datos.

Acerca de esta tarea

Un programa anidado también puede acceder a elementos de datos en un programa hermano (uno en el mismo nivel de anidamiento en el mismo programa contenedor) que se declara con el atributo `COMMON` .

Conceptos relacionados
“Programas anidados” en la página 459

Compartir datos en programas recursivos

Si el programa tiene el atributo RECURSIVE, los datos definidos en LINKAGE SECTION no son accesibles en las invocaciones posteriores del programa.

Acerca de esta tarea

Para direccionar un registro en LINKAGE SECTION, utilice cualquiera de estas técnicas:

- Pase un argumento al programa y especifique el registro en una posición adecuada en la frase USING del programa.
- Utilice la sentencia format-5 SET .

Si el programa tiene el atributo RECURSIVE, la dirección del registro es válida para una instancia determinada de la invocación del programa. La dirección del registro en otra instancia de ejecución del mismo programa debe restablecerse para dicha instancia de ejecución. Se producirán resultados imprevisibles si hace referencia a un elemento de datos para el que no se ha establecido la dirección.

Tareas relacionadas
"Realización de llamadas recursivas" en la página 470

Referencias relacionadas

sentencia SET (*COBOL for Linux en x86 Consulta de lenguaje*)

Proceso de los datos

En el PROCEDURE DIVISION de un programa, codifica las sentencias ejecutables que procesan los datos que ha definido en las otras divisiones. PROCEDURE DIVISION contiene una o dos cabeceras y la lógica del programa.

Acerca de esta tarea

El PROCEDURE DIVISION empieza con la cabecera de división y una cabecera de nombre de procedimiento. La cabecera de división de un programa puede ser simplemente:

```
PROCEDURE DIVISION.
```

Puede codificar la cabecera de división para recibir parámetros utilizando la frase USING o para devolver un valor utilizando la frase RETURNING .

Para recibir un argumento que se ha pasado por referencia (el valor predeterminado) o por contenido, codifique la cabecera de división para un programa de una de estas formas:

```
PROCEDURE DIVISION USING dataname
PROCEDURE DIVISION USING BY REFERENCE dataname
```

Asegúrese de definir *dataname* en LINKAGE SECTION de DATA DIVISION.

Para recibir un parámetro que se ha pasado por valor, codifique la cabecera de división para un programa como se indica a continuación:

```
PROCEDURE DIVISION USING BY VALUE dataname
```

Para devolver un valor como resultado, codifique la cabecera de división como se indica a continuación:

```
PROCEDURE DIVISION RETURNING dataname2
```

También puede combinar USING y RETURNING en una cabecera PROCEDURE DIVISION :

```
PROCEDURE DIVISION USING dataname RETURNING dataname2
```

Asegúrese de definir *dataname* y *dataname2* en LINKAGE SECTION.

Conceptos relacionados
“Cómo se divide la lógica en PROCEDURE DIVISION” en la página 15

Tareas relacionadas
“Codificación de la SECCIÓN DE ENLACE” en la página 476
“Codificación de PROCEDURE DIVISION para pasar argumentos” en la página 477
“Utilizando PROCEDURE DIVISION REGRESE...” en la página 482
“Eliminación de codificación repetitiva” en la página 537

Referencias relacionadas
La cabecera de división de procedimiento (*COBOL for Linux en x86 Consulta de lenguaje*)
La frase USING (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia CALL (*COBOL for Linux en x86 Consulta de lenguaje*)

Cómo se divide la lógica en PROCEDURE DIVISION

El PROCEDURE DIVISION de un programa se divide en secciones y párrafos, que contienen frases, sentencias y frases.

Sección
Subdivisión lógica de la lógica de proceso.

Una sección tiene una cabecera de sección y, opcionalmente, va seguida de uno o varios párrafos.

Una sección puede ser objeto de una sentencia PERFORM . Un tipo de sección es para declarativos.

Párrafo
Subdivisión de una sección, procedimiento o programa.

Un párrafo tiene un nombre seguido de un punto y cero o más frases.

Un párrafo puede ser objeto de una declaración.

Frase
Serie de una o más sentencias COBOL que finaliza con un punto.

Declaración
Realiza un paso definido del proceso COBOL, como por ejemplo la adición de dos números.

Una sentencia es una combinación válida de palabras y empieza por una sentencia COBOL. Las sentencias son imperativas (que indican una acción incondicional), condicionales o que dirigen el compilador. Se prefiere utilizar terminadores de ámbito explícitos en lugar de puntos para mostrar el final lógico de una sentencia.

Frase
Subdivisión de una sentencia.

Conceptos relacionados
“Sentencias de direccionamiento de compilador” en la página 17
“Terminadores de ámbito” en la página 17
“Declaraciones imperativas” en la página 16
“Sentencias condicionales” en la página 16
“Declaraciones” en la página 18

Referencias relacionadas
Estructura PROCEDURE DIVISION (*COBOL for Linux en x86 Consulta de lenguaje*)

Declaraciones imperativas

Una sentencia imperativa (como ADD, MOVE, CALLo CLOSE) indica que se debe realizar una acción incondicional.

Puede finalizar una sentencia imperativa con un terminador de ámbito implícito o explícito.

Una sentencia condicional que finaliza con un terminador de ámbito explícito se convierte en una sentencia imperativa denominada *sentencia de ámbito delimitado*. Sólo se pueden anidar sentencias imperativas (o sentencias de ámbito delimitado).

Conceptos relacionados
"Sentencias condicionales" en la página 16
"Terminadores de ámbito" en la página 17

Sentencias condicionales

Una sentencia condicional es una sentencia condicional simple (IF, EVALUATE, SEARCH) o una sentencia condicional formada por una sentencia imperativa que incluye una frase u opción condicional.

Puede finalizar una sentencia condicional con un terminador de ámbito implícito o explícito. Si finaliza una sentencia condicional explícitamente, se convierte en una sentencia de ámbito delimitada (que es una sentencia imperativa).

Puede utilizar una sentencia de ámbito delimitado de estas maneras:

- Para delimitar el rango de operación para una sentencia condicional COBOL y para mostrar explícitamente los niveles de anidamiento

  Por ejemplo, utilice una frase END-IF en lugar de un punto para finalizar el ámbito de una sentencia IF dentro de un IFanidado.
- Para codificar una sentencia condicional en la que la sintaxis COBOL llama a una sentencia imperativa

  Por ejemplo, codifique una sentencia condicional como el objeto de un PERFORMen línea:

```
PERFORM UNTIL TRANSACTION-EOF
    PERFORM 200-EDIT-UPDATE-TRANSACTION
    IF NO-ERRORS
        PERFORM 300-UPDATE-COMMUTER-RECORD
    ELSE
        PERFORM 400-PRINT-TRANSACTION-ERRORS
    END-IF
    READ UPDATE-TRANSACTION-FILE INTO WS-TRANSACTION-RECORD
        AT END
            SET TRANSACTION-EOF TO TRUE
    END-READ
END-PERFORM
```

  Se necesita un terminador de ámbito explícito para la sentencia PERFORM en línea, pero no es válido para la sentencia PERFORM fuera de línea.

Para un control de programa adicional, puede utilizar la frase NOT con sentencias condicionales. Por ejemplo, puede proporcionar instrucciones que se deben realizar cuando no se produce una excepción determinada, como por ejemplo NOT ON SIZE ERROR. La frase NOT no se puede utilizar con la frase ON OVERFLOW de la sentencia CALL , pero se puede utilizar con la frase ON EXCEPTION .

No anide sentencias condicionales. Las sentencias anidadas deben ser sentencias imperativas (o sentencias de ámbito delimitado) y deben seguir las reglas para sentencias imperativas.

Las sentencias siguientes son ejemplos de sentencias condicionales si están codificadas sin terminadores de ámbito:

- Sentencia aritmética con ON SIZE ERROR
- Sentencias de manipulación de datos con ON OVERFLOW

- Sentencias CALL con ON OVERFLOW
- Sentencias I/O con INVALID KEY, AT ENDo AT END-OF-PAGE
- RETURN con AT END

Conceptos relacionados
"Declaraciones imperativas" en la página 16
"Terminadores de ámbito" en la página 17

Tareas relacionadas
"Selección de acciones de programa" en la página 83

Referencias relacionadas
Sentencias condicionales (*COBOL for Linux en x86 Consulta de lenguaje*)

Sentencias de direccionamiento de compilador

Una sentencia de dirección de compilador hace que el compilador realice una acción específica sobre la estructura del programa, COPY el proceso, el control de listado, flujo de control o CALL convenio de interfaz.

Una sentencia de dirección de compilador no forma parte de la lógica del programa.

Referencias relacionadas
Capítulo 14, "Sentencias de direccionamiento de compilador", en la página 309
Sentencias de direccionamiento de compilador (*COBOL for Linux en x86 Consulta de lenguaje*)

Terminadores de ámbito

Un terminador de ámbito finaliza una sentencia. Los terminadores de ámbito pueden ser explícitos o implícitos.

Los terminadores de ámbito explícitos finalizan una sentencia sin finalizar una frase. Constan de END seguido de un guión y el nombre de la sentencia que se está terminando, como por ejemplo END-IF. Un terminador de ámbito implícito es un punto (.) que finaliza el ámbito de todas las sentencias anteriores que todavía no han finalizado.

Cada uno de los dos puntos del siguiente fragmento de programa finaliza una sentencia IF , lo que hace que el código sea equivalente al código posterior que utiliza terminadores de ámbito explícitos:

```
IF ITEM = "A"
    DISPLAY "THE VALUE OF ITEM IS " ITEM
    ADD 1 TO TOTAL
    MOVE "C" TO ITEM
    DISPLAY "THE VALUE OF ITEM IS NOW " ITEM.
IF ITEM = "B"
    ADD 2 TO TOTAL.
```

```
IF ITEM = "A"
    DISPLAY "THE VALUE OF ITEM IS " ITEM
    ADD 1 TO TOTAL
    MOVE "C" TO ITEM
    DISPLAY "THE VALUE OF ITEM IS NOW " ITEM
END-IF
IF ITEM = "B"
    ADD 2 TO TOTAL
END-IF
```

Si utiliza terminadores implícitos, el final de las sentencias puede no estar claro. Como resultado, puede finalizar las sentencias de forma involuntaria, cambiando la lógica del programa. Los terminadores de ámbito explícitos hacen que un programa sea más fácil de entender e impiden la finalización involuntaria

de las sentencias. Por ejemplo, en el fragmento de programa siguiente, al cambiar la ubicación del primer periodo en el primer ámbito implícito, el ejemplo cambia el significado del código:

```
IF ITEM = "A"
    DISPLAY "VALUE OF ITEM IS " ITEM
    ADD 1 TO TOTAL.
    MOVE "C" TO ITEM
    DISPLAY " VALUE OF ITEM IS NOW " ITEM
IF ITEM = "B"
    ADD 2 TO TOTAL.
```

La sentencia MOVE y la sentencia DISPLAY después se realizan independientemente del valor de ITEM, a pesar de lo que indica la sangría, porque el primer punto termina la sentencia IF .

Para mejorar la claridad del programa y evitar la finalización no intencionada de las sentencias, utilice terminadores de ámbito explícitos, especialmente dentro de los párrafos. Utilice terminadores de ámbito implícitos sólo al final de un párrafo o al final de un programa.

Tenga cuidado al codificar un terminador de ámbito explícito para una sentencia imperativa anidada en una sentencia condicional. Asegúrese de que el terminador de ámbito esté emparejado con la sentencia para la que estaba previsto. En el ejemplo siguiente, el terminador de ámbito se emparejará con la segunda sentencia READ , aunque el programador pretendía que se emparejara con la primera.

```
READ FILE1
  AT END
    MOVE A TO B
    READ FILE2
END-READ
```

Para asegurarse de que el terminador de ámbito explícito se empareja con la sentencia prevista, el ejemplo anterior se puede recodificar de este modo:

```
READ FILE1
  AT END
    MOVE A TO B
    READ FILE2
    END-READ
END-READ
```

Conceptos relacionados
“Sentencias condicionales” en la página 16
“Declaraciones imperativas” en la página 16

Declaraciones

Los declarativos proporcionan una o más secciones de propósito especial que se ejecutan cuando se produce una condición de excepción.

Inicie cada sección declarativa con una sentencia USE que identifique la función de la sección. En los procedimientos, especifique las acciones que deben llevarse a cabo cuando se produzca la condición.

Tareas relacionadas
“Búsqueda y manejo de errores de entrada-salida” en la página 320

Referencias relacionadas
Declaratives (*COBOL for Linux en x86 Consulta de lenguaje*)

Capítulo 2. Utilización de datos

Acerca de esta tarea

Esta información está pensada para ayudar a los programadores no COBOL a relacionar términos para datos utilizados en otros lenguajes de programación con términos COBOL. Presenta fundamentos COBOL para variables, estructuras, literales y constantes; asignación y visualización de valores; funciones intrínsecas (incorporadas) y tablas (matrices) y punteros.

Tareas relacionadas
"Utilización de variables, estructuras, literales y constantes" en la página 19
"Asignación de valores a elementos de datos" en la página 23
"Visualización de valores en una pantalla o en un archivo (DISPLAY)" en la página 32
"Utilización de funciones intrínsecas (funciones incorporadas)" en la página 33
"Utilización de tablas (matrices) y punteros" en la página 34
Capítulo 10, "Tratamiento de datos en un entorno internacional", en la página 189

Utilización de variables, estructuras, literales y constantes

La mayoría de los lenguajes de programación de alto nivel comparten el concepto de que los datos se representan como variables, estructuras (elementos de grupo), literales o constantes.

Acerca de esta tarea

Los datos de un programa COBOL pueden ser alfabéticos, alfanuméricos, juego de caracteres de doble byte (DBCS), nacional o numérico. También puede definir nombres de índice y elementos de datos descritos como `USAGE POINTER`, `USAGE FUNCTION-POINTER`, o `USAGE PROCEDURE-POINTER`. Coloque todas las definiciones de datos en el `DATA DIVISION` del programa.

Tareas relacionadas
"Utilización de variables" en la página 19
"Utilización de elementos de datos y elementos de grupo" en la página 20
"Utilización de literales" en la página 21
"Utilización de constantes" en la página 22
"Utilización de constantes figurativas" en la página 22

Referencias relacionadas
Clases y categorías de datos (*COBOL for Linux en x86 Consulta de lenguaje*)

Utilización de variables

Una *variable* es un elemento de datos cuyo valor puede cambiar durante un programa. Sin embargo, el valor está restringido al tipo de datos que defina cuando especifique un nombre y una longitud para el elemento de datos.

Acerca de esta tarea

Por ejemplo, si un nombre de cliente es un elemento de datos alfanumérico en el programa, puede definir y utilizar el nombre de cliente tal como se muestra a continuación:

```
Data Division.
01  Customer-Name           Pic X(20).
01  Original-Customer-Name  Pic X(20).
. . .
Procedure Division.
```

```
    Move Customer-Name to Original-Customer-Name
    . . .
```

En su lugar, puede definir los nombres de cliente anteriores como elementos de datos nacionales especificando sus cláusulas PICTURE como `Pic N(20)` y especificando la cláusula USAGE NATIONAL para los elementos. Los elementos de datos nacionales se representan en Unicode UTF-16, en el que la mayoría de los caracteres se representan en 2 bytes de almacenamiento.

Conceptos relacionados
"Unicode y la codificación de caracteres de idioma" en la página 190

Tareas relacionadas
"Utilización de datos nacionales (Unicode) en COBOL" en la página 191

Referencias relacionadas
"SÍMBOLO" en la página 290
"Almacenamiento de datos de caracteres" en la página 205
Cláusula PICTURE (*COBOL for Linux en x86 Consulta de lenguaje*)

Utilización de elementos de datos y elementos de grupo

Los elementos de datos relacionados pueden ser partes de una estructura de datos jerárquica. Un elemento de datos que no tiene elementos de datos subordinados se denomina *elemento elemental*. Un elemento de datos que se compone de uno o más elementos de datos subordinados se denomina *elemento de grupo*.

Acerca de esta tarea

Un registro puede ser un elemento elemental o un elemento de grupo. Un elemento de grupo puede ser un *elemento de grupo alfanumérico* o un *elemento de grupo nacional*.

Por ejemplo, `Customer-Record` a continuación es un elemento de grupo alfanumérico que se compone de dos elementos de grupo alfanuméricos subordinados (`Customer-Name` y `Part-Order`), cada uno de los cuales contiene elementos de datos elementales. Estos elementos de grupo tienen implícitamente USAGE DISPLAY. Puede hacer referencia a un elemento de grupo completo o a partes de un elemento de grupo en sentencias MOVE en la PROCEDURE DIVISION tal como se muestra a continuación:

```
Data Division.
File Section.
FD  Customer-File
    Record Contains 45 Characters.
01  Customer-Record.
    05  Customer-Name.
        10 Last-Name        Pic x(17).
        10 Filler           Pic x.
        10 Initials         Pic xx.
    05  Part-Order.
        10 Part-Name        Pic x(15).
        10 Part-Color       Pic x(10).
Working-Storage Section.
01  Orig-Customer-Name.
    05  Surname            Pic x(17).
    05  Initials           Pic x(3).
01  Inventory-Part-Name     Pic x(15).
. . .
Procedure Division.
    Move Customer-Name to Orig-Customer-Name
    Move Part-Name to Inventory-Part-Name
    . . .
```

En su lugar, puede definir `Customer-Record` como un elemento de grupo nacional que se compone de dos elementos de grupo nacional subordinados cambiando las declaraciones en la DATA DIVISION tal como se muestra a continuación. Los elementos de grupo nacionales se comportan de la misma manera que los elementos de datos nacionales de categoría elemental en la mayoría de las operaciones. La cláusula GROUP-USAGE NATIONAL indica que un elemento de grupo y cualquier elemento de grupo

subordinado al mismo son grupos nacionales. Los elementos elementales subordinados de un grupo nacional deben describirse explícita o implícitamente como USAGE NATIONAL.

```
Data Division.
File Section.
FD  Customer-File
    Record Contains 90 Characters.
01  Customer-Record         Group-Usage National.
    05  Customer-Name.
        10 Last-Name        Pic n(17).
        10 Filler           Pic n.
        10 Initials         Pic nn.
    05  Part-Order.
        10 Part-Name        Pic n(15).
        10 Part-Color       Pic n(10).
Working-Storage Section.
01  Orig-Customer-Name      Group-Usage National.
    05  Surname            Pic n(17).
    05  Initials           Pic n(3).
01  Inventory-Part-Name     Pic n(15)  Usage National.
. . .
Procedure Division.
    Move Customer-Name to Orig-Customer-Name
    Move Part-Name to Inventory-Part-Name
    . . .
```

En el ejemplo anterior, los elementos de grupo podrían especificar la cláusula USAGE NATIONAL a nivel de grupo. Una cláusula USAGE a nivel de grupo se aplica a cada elemento de datos elemental de un grupo (y, por lo tanto, sirve como una notación abreviada conveniente). Sin embargo, un grupo que especifica la cláusula USAGE NATIONAL *no* es un grupo nacional a pesar de la representación de los elementos elementales dentro del grupo. Los grupos que especifican la cláusula USAGE son grupos alfanuméricos y se comportan en muchas operaciones, como movimientos y comparaciones, como elementos de datos elementales de USAGE DISPLAY (excepto que no se produce ninguna edición o conversión de datos).

Conceptos relacionados
“Unicode y la codificación de caracteres de idioma” en la página 190
“Grupos nacionales” en la página 198

Tareas relacionadas
“Utilización de datos nacionales (Unicode) en COBOL” en la página 191
“Utilización de grupos nacionales” en la página 202

Referencias relacionadas
“Entradas de FILE SECTION” en la página 10
“Almacenamiento de datos de caracteres” en la página 205
Clases y categorías de elementos de grupo (*COBOL for Linux en x86 Consulta de lenguaje*)
Cláusula PICTURE (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia MOVE (*COBOL for Linux en x86 Consulta de lenguaje*)
Cláusula USAGE (*COBOL for Linux en x86 Consulta de lenguaje*)

Utilización de literales

Un *literal* es una serie de caracteres cuyo valor se proporciona mediante los propios caracteres. Si conoce el valor que desea que tenga un elemento de datos, puede utilizar una representación literal del valor de datos en PROCEDURE DIVISION.

Acerca de esta tarea

No es necesario definir un elemento de datos para el valor ni hacer referencia a él utilizando un nombre de datos. Por ejemplo, puede preparar un mensaje de error para un archivo de salida moviendo un literal alfanumérico:

```
Move "Name is not valid" To Customer-Name
```

Puede comparar un elemento de datos con un valor entero específico utilizando un literal numérico. En el ejemplo siguiente, "Name is not valid" es un literal alfanumérico y 03519 es un literal numérico:

```
01  Part-number     Pic 9(5).
. . .
    If Part-number = 03519 then display "Part number was found"
```

Puede utilizar el formato de notación hexadecimal (X') para expresar los caracteres de control X'00' a X'1F' dentro de un literal alfanumérico. Los resultados son imprevisibles si especifica estos caracteres de control en el formato básico de literales alfanuméricos.

Puede utilizar el delimitador de apertura N" o N' para designar un literal nacional si la opción de compilador NSYMBOL(NATIONAL) está en vigor, o para designar un literal DBCS si la opción de compilador NSYMBOL(DBCS) está en vigor.

Puede utilizar el delimitador de apertura NX" o NX' para designar literales nacionales en notación hexadecimal (independientemente del valor de la opción de compilador NSYMBOL). Cada grupo de cuatro dígitos hexadecimales designa un único carácter nacional.

Conceptos relacionados
“Unicode y la codificación de caracteres de idioma” en la página 190

Tareas relacionadas
“Utilización de literales nacionales” en la página 192
“Utilización de literales DBCS” en la página 211

Referencias relacionadas
“SÍMBOLO” en la página 290
Literales (*COBOL for Linux en x86 Consulta de lenguaje*)

Utilización de constantes

Una *constante* es un elemento de datos que sólo tiene un valor. COBOL no define una construcción para constantes. Sin embargo, puede definir un elemento de datos con un valor inicial codificando una cláusula VALUE en la descripción de datos (en lugar de codificar una sentencia INITIALIZE).

Acerca de esta tarea

```
Data Division.
01  Report-Header   pic x(50)  value "Company Sales Report".
. . .
01  Interest        pic 9v9999 value 1.0265.
```

El ejemplo anterior inicializa un elemento de datos alfanumérico y numérico. Asimismo, puede utilizar una cláusula VALUE para definir una constante nacional o DBCS.

Tareas relacionadas
“Utilización de datos nacionales (Unicode) en COBOL” en la página 191
“Codificación para el uso del soporte DBCS” en la página 210

Utilización de constantes figurativas

Determinadas constantes y literales utilizados habitualmente están disponibles como palabras reservadas denominadas *constantes figurativas*: ZERO, SPACE, HIGH-VALUE, LOW-VALUE, QUOTE, NULLy ALL *literal*. Puesto que representan valores fijos, las constantes figurativas no requieren una definición de datos.

Acerca de esta tarea

Por ejemplo:

```
Move Spaces To Report-Header
```

Tareas relacionadas
“Utilización de constantes figurativas de carácter nacional” en la página 196
“Codificación para el uso del soporte DBCS” en la página 210

Referencias relacionadas
Constantes figurativas (*COBOL for Linux en x86 Consulta de lenguaje*)

Asignación de valores a elementos de datos

Después de haber definido un elemento de datos, puede asignarle un valor en cualquier momento. La asignación adopta muchas formas en COBOL, en función de lo que desee hacer.

Acerca de esta tarea

Tabla 2. ***Asignación a elementos de datos en un programa***

Qué desea hacer	**Cómo hacerlo**
Asignar valores a un elemento de datos o área de datos grande.	Utilice una de estas formas: • Sentencia `INITIALIZE` • Sentencia `MOVE` • Sentencia `STRING` o `UNSTRING` • Cláusula `VALUE` (para establecer elementos de datos en los valores que desea que tengan cuando el programa está en estado inicial)
Asigne los resultados de la aritmética.	Utilice sentencias `COMPUTE`, `ADD`, `SUBTRACT`, `MULTIPLY`o `DIVIDE` .
Examine o sustituya caracteres o grupos de caracteres en un elemento de datos.	Utilice la sentencia `INSPECT` .
Recibir valores de un archivo.	Utilice la sentencia `READ` (o `READ INTO`).
Recibir valores de un dispositivo de entrada del sistema o un archivo.	Utilice la sentencia `ACCEPT` .
Establezca una constante.	Utilice la cláusula `VALUE` en la definición del elemento de datos y no utilice el elemento de datos como receptor. Un elemento de este tipo es en efecto una constante aunque el compilador no aplique constantes de sólo lectura.

*Tabla 2. **Asignación a elementos de datos en un programa** (continuación)*	
Qué desea hacer	**Cómo hacerlo**
Una de estas acciones: • Coloque un valor asociado con un elemento de tabla en un índice. • Establezca el estado de un conmutador externo en ON o OFF. • Mueva los datos a un nombre de condición para que la condición sea verdadera. • Establezca un elemento de datos POINTER, PROCEDURE-POINTERo FUNCTION-POINTER en una dirección.	Utilice la sentencia SET .

"Ejemplos: inicialización de elementos de datos" en la página 24

Tareas relacionadas

"Inicialización de una estructura (INITIALIZE)" en la página 27
"Asignación de valores a elementos de datos elementales (MOVE)" en la página 28
"Asignación de valores a elementos de datos de grupo (MOVE)" en la página 29
"Asignación de entrada desde una pantalla o archivo (ACCEPT)" en la página 31
"Unión de elementos de datos (STRING)" en la página 95
"División de elementos de datos (UNSTRING)" en la página 97
"Asignación de resultados aritméticos (MOVE o COMPUTE)" en la página 30
"Recuento y sustitución de elementos de datos (INSPECT)" en la página 104
Capítulo 10, "Tratamiento de datos en un entorno internacional", en la página 189

Ejemplos: inicialización de elementos de datos

Los ejemplos siguientes muestran cómo puede inicializar muchos tipos de elementos de datos, incluidos los elementos de datos alfanuméricos, editados a nivel nacional y editados a nivel numérico, utilizando sentencias INITIALIZE .

Una sentencia INITIALIZE es funcionalmente equivalente a una o más sentencias MOVE . Las tareas relacionadas sobre la inicialización muestran cómo puede utilizar una sentencia INITIALIZE en un elemento de grupo para inicializar convenientemente todos los elementos de datos subordinados que están en una categoría de datos determinada.

Inicialización de un elemento de datos en blancos o ceros:

```
INITIALIZE identifier-1
```

***identifier-1* PICTURE**	***identifier-1* antes**	***identifier-1* después**
9(5)	12345	00000
X(5)	AB123	*bbbbb*[1]
N(3)	410042003100[2]	200020002000[3]
99XX9	12AB3	*bbbbb*[1]
XXBX/XX	ABbC/DE	*bbbb/bb*[1]
**99.9CR	1234.5CR	**00.0*bb*[1]
A(5)	ABCDE	*bbbbb*[1]
+99.99E+99	+12.34E+02	+00.00E+00

identifier-1 **PICTURE**	*identifier-1* **antes**	*identifier-1* **después**
1. El símbolo *b* representa un espacio en blanco. 2. Representación hexadecimal de los caracteres nacionales (UTF-16) 'AB1'. El ejemplo presupone que *identifier-1* tiene `Usage National`. 3. Representación hexadecimal de los caracteres nacionales (UTF-16) ' ' (tres espacios en blanco). Tenga en cuenta que si *identifier-1* no se definiera como `Usage National`, y si `NSYMBOL(DBCS)` estuviera en vigor, `INITIALIZE` almacenaría en su lugar los espacios DBCS ('2020') en *identifier-1*.		

Inicialización de un elemento de datos alfanumérico:

```
01  ALPHANUMERIC-1    PIC X    VALUE "y".
01  ALPHANUMERIC-3    PIC X(1) VALUE "A".
. . .
    INITIALIZE ALPHANUMERIC-1
        REPLACING ALPHANUMERIC DATA BY ALPHANUMERIC-3
```

ALPHANUMERIC-3	**ALPHANUMERIC-1 antes**	**ALPHANUMERIC-1 después de**
A	y	A

Inicialización de un elemento de datos alfanumérico justificado por la derecha:

```
01  ANJUST            PIC X(8)  VALUE SPACES  JUSTIFIED RIGHT.
01  ALPHABETIC-1      PIC A(4)  VALUE "ABCD".
. . .
    INITIALIZE ANJUST
        REPLACING ALPHANUMERIC DATA BY ALPHABETIC-1
```

ALPHABETIC-1	**ANJUST antes**	**ANJUST después de**
ABCD	*bbbbbbbb*[1]	*bbbb*ABCD[1]
1. El símbolo *b* representa un espacio en blanco.		

Inicialización de un elemento de datos editado alfanumérico:

```
01  ALPHANUM-EDIT-1   PIC XXBX/XXX  VALUE "ABbC/DEF".
01  ALPHANUM-EDIT-3   PIC X/BB      VALUE "M/bb".
. . .
    INITIALIZE ALPHANUM-EDIT-1
        REPLACING ALPHANUMERIC-EDITED DATA BY ALPHANUM-EDIT-3
```

ALPHANUM-EDIT-3	**ALPHANUM-EDIT-1 antes**	**ALPHANUM-EDIT-1 después de**
M/*bb*[1]	AB*b*C/DEF[1]	M/*bb*/*bbb*[1]
1. El símbolo *b* representa un espacio en blanco.		

Inicialización de un elemento de datos nacional:

```
01  NATIONAL-1        PIC NN  USAGE NATIONAL  VALUE N"AB".
01  NATIONAL-3        PIC NN  USAGE NATIONAL  VALUE N"CD".
. . .
    INITIALIZE NATIONAL-1
        REPLACING NATIONAL DATA BY NATIONAL-3
    INITIALIZE NATIONAL-1 NATIONAL TO VALUE
```

NATIONAL-3	NATIONAL-1 antes de la primera INITIALIZE	NATIONAL-1 después del primer INITIALIZE	NATIONAL-1 después del segundo INITIALIZE
43004400[1]	41004200[2]	43004400[1]	41004200

1. Representación hexadecimal de los caracteres nacionales 'CD'
2. Representación hexadecimal de los caracteres nacionales 'AB'

Inicialización de un elemento de datos editado a nivel nacional:

```
01  NATL-EDIT-1       PIC 0NN  USAGE NATIONAL  VALUE N"123".
01  NATL-3            PIC NNN  USAGE NATIONAL  VALUE N"456".
. . .
    INITIALIZE NATL-EDIT-1
        REPLACING NATIONAL-EDITED DATA BY NATL-3
```

NATL-3	NATL-EDIT-1 antes	NATL-EDIT-1 después de
340035003600[1]	310032003300[2]	300034003500[3]

1. Representación hexadecimal de los caracteres nacionales '456'
2. Representación hexadecimal de los caracteres nacionales '123'
3. Representación hexadecimal de los caracteres nacionales '045'

Inicialización de un elemento de datos numérico (decimal con zona):

```
01  NUMERIC-1         PIC 9(8)        VALUE 98765432.
01  NUM-INT-CMPT-3    PIC 9(7)  COMP  VALUE 1234567.
. . .
    INITIALIZE NUMERIC-1
        REPLACING NUMERIC DATA BY NUM-INT-CMPT-3
```

NUM-INT-CMPT-3	NUMERIC-1 antes	NUMERIC-1 después de
1234567	98765432	01234567

Inicialización de un elemento de datos numérico (decimal nacional):

```
01  NAT-DEC-1         PIC 9(3)  USAGE  NATIONAL VALUE 987.
01  NUM-INT-BIN-3     PIC 9(2)  BINARY VALUE 12.
. . .
    INITIALIZE NAT-DEC-1
        REPLACING NUMERIC DATA BY NUM-INT-BIN-3
```

NUM-INT-BIN-3	NAT-DEC-1 antes	NAT-DEC-1 después de
12	390038003700[1]	300031003200[2]

1. Representación hexadecimal de los caracteres nacionales '987'
2. Representación hexadecimal de los caracteres nacionales '012'

Inicialización de un elemento de datos editado numérica-editado (USAGE DISPLAY):

```
01  NUM-EDIT-DISP-1   PIC $ZZ9V  VALUE "$127".
01  NUM-DISP-3        PIC 999V   VALUE 12.
. . .
```

```
    INITIALIZE NUM-EDIT-DISP-1
        REPLACING NUMERIC-EDITED DATA BY NUM-DISP-3
```

NUM-DISP-3	NUM-EDIT-DISP-1 antes	NUM-EDIT-DISP-1 después de
012	$127	$ 12

Inicialización de un elemento de datos editado numérica-editado (USAGE NATIONAL):

```
01  NUM-EDIT-NATL-1   PIC $ZZ9V  NATIONAL VALUE N"$127".
01  NUM-NATL-3       PIC 999V   NATIONAL VALUE 12.
. . .
    INITIALIZE NUM-EDIT-NATL-1
        REPLACING NUMERIC-EDITED DATA BY NUM-NATL-3
```

NUM-NATL-3	NUM-EDIT-NATL-1 antes	NUM-EDIT-NATL-1 después de
300031003200[1]	2400310032003700[2]	2400200031003200[3]

1. Representación hexadecimal de los caracteres nacionales '012'
2. Representación hexadecimal de los caracteres nacionales '$127'
3. Representación hexadecimal de los caracteres nacionales '$ 12'

Tareas relacionadas
“Inicialización de una estructura (INITIALIZE)” en la página 27
“Inicialización de una tabla (INITIALIZE)” en la página 65
“Definición de datos numéricos” en la página 35

Referencias relacionadas
“SÍMBOLO” en la página 290

Inicialización de una estructura (INITIALIZE)

Puede restablecer los valores de todos los elementos de datos subordinados de un elemento de grupo aplicando la sentencia INITIALIZE a ese elemento de grupo. Sin embargo, es ineficiente inicializar un grupo completo a menos que realmente necesite que se inicialicen todos los elementos del grupo.

Acerca de esta tarea

El ejemplo siguiente muestra cómo puede restablecer los campos a espacios y ceros en los registros de transacción que produce un programa. Los valores de los campos no son idénticos en cada registro que se genera. (El registro de transacción se define como un elemento de grupo alfanumérico, TRANSACTION-OUT.)

```
01  TRANSACTION-OUT.
    05  TRANSACTION-CODE        PIC X.
    05  PART-NUMBER             PIC 9(6).
    05  TRANSACTION-QUANTITY    PIC 9(5).
    05  PRICE-FIELDS.
        10  UNIT-PRICE          PIC 9(5)V9(2).
        10  DISCOUNT            PIC V9(2).
        10  SALES-PRICE         PIC 9(5)V9(2).
. . .
    INITIALIZE TRANSACTION-OUT
```

Registro	TRANSACTION-OUT antes	TRANSACTION-OUT después de
1	R00138300024000000000000000	*b*00000000000000000000000000[1]
2	R00139000048000000000000000	*b*00000000000000000000000000[1]

Registro	TRANSACTION-OUT antes	TRANSACTION-OUT después de
3	S00141000012000000000000000	*b*00000000000000000000000000[1]
4	C00138300000000425000000000	*b*00000000000000000000000000[1]
5	C00201000000000000100000000	*b*00000000000000000000000000[1]
1. El símbolo *b* representa un espacio en blanco.		

De la misma forma, puede restablecer los valores de todos los elementos de datos subordinados en un elemento de grupo nacional aplicando la sentencia INITIALIZE a dicho elemento de grupo. La estructura siguiente es similar a la estructura anterior, pero en su lugar utiliza datos Unicode UTF-16 :

```
01  TRANSACTION-OUT GROUP-USAGE NATIONAL.
    05  TRANSACTION-CODE        PIC N.
    05  PART-NUMBER             PIC 9(6).
    05  TRANSACTION-QUANTITY    PIC 9(5).
    05  PRICE-FIELDS.
        10  UNIT-PRICE          PIC 9(5)V9(2).
        10  DISCOUNT            PIC V9(2).
        10  SALES-PRICE         PIC 9(5)V9(2).
. . .
    INITIALIZE TRANSACTION-OUT
```

Independientemente del contenido anterior del registro de transacción, después de ejecutar la sentencia INITIALIZE anterior:

- TRANSACTION-CODE contiene NX"2000" (un espacio nacional).
- Cada una de las 27 posiciones de caracteres nacionales restantes de TRANSACTION-OUT contiene NX"3000" (un cero decimal nacional).

Cuando se utiliza una sentencia INITIALIZE para inicializar un elemento de datos de grupo alfanumérico o nacional, el elemento de datos se procesa como un elemento de grupo, es decir, con semántica de grupo. Los elementos de datos elementales del grupo se reconocen y procesan, tal como se muestra en los ejemplos anteriores. Si no codifica la frase REPLACING de la sentencia INITIALIZE :

- SPACE es el elemento emisor implícito para los elementos receptores alfabéticos, alfanuméricos, editados alfanuméricos, DBCS, de categoría nacional y editados a nivel nacional.
- ZERO es el elemento de envío implícito para los elementos de recepción numéricos y numéricos editados.

Conceptos relacionados
“Grupos nacionales” en la página 198

Tareas relacionadas
“Inicialización de una tabla (INITIALIZE)” en la página 65
“Utilización de grupos nacionales” en la página 202

Referencias relacionadas
sentencia INITIALIZE (*COBOL for Linux en x86 Consulta de lenguaje*)

Asignación de valores a elementos de datos elementales (MOVE)

Utilice una sentencia MOVE para asignar un valor a un elemento de datos elemental.

Acerca de esta tarea

La sentencia siguiente asigna el contenido de un elemento de datos elemental , `Customer-Name`, al elemento de datos elemental `Orig-Customer-Name`:

```
Move Customer-Name to Orig-Customer-Name
```

Si `Customer-Name` es más largo que `Orig-Customer-Name`, el truncamiento se produce a la derecha. Si `Customer-Name` es más corto, las posiciones de caracteres adicionales a la derecha en `Orig-Customer-Name` se rellenan con espacios.

Para los elementos de datos que contienen números, los movimientos pueden ser más complicados que con los elementos de datos de caracteres porque hay varias formas en las que se pueden representar los números. En general, los valores algebraicos de los números se mueven si es posible, en contraposición a los movimientos dígito por dígito que se realizan con los datos de caracteres. Por ejemplo, después de la sentencia MOVE siguiente, `Item-x` contiene el valor 3.0, representado como 0030:

```
01  Item-x      Pic 999v9.
. . .
    Move 3.06 to Item-x
```

Puede mover un elemento de datos alfabético, alfanumérico, editado alfanumérico, DBCS, entero o editado numérico a un elemento de datos de categoría nacional o editado nacional; el elemento de envío se convierte. Puede mover un elemento de datos nacional a un elemento de datos de categoría nacional o editado a nivel nacional de . Si el contenido de un elemento de datos nacional de categoría tiene un valor numérico, puede mover dicho elemento a un elemento de datos numérico, editado numérico, de coma flotante externo o de coma flotante interno. Puede mover un elemento de datos editado a nivel nacional sólo a un elemento de datos nacional de categoría u otro elemento de datos editado a nivel nacional. Es posible que se produzca un relleno o truncamiento.

Para obtener detalles completos sobre movimientos elementales, consulte la referencia relacionada a continuación sobre la sentencia MOVE .

El ejemplo siguiente muestra un elemento de datos alfanumérico en el idioma griego que se mueve a un elemento de datos nacional:

```
. . .
01 Data-in-Unicode   Pic N(100) usage national.
01 Data-in-Greek     Pic X(100).
. . .
    Read Greek-file into Data-in-Greek
    Move Data-in-Greek to Data-in-Unicode
```

Conceptos relacionados
“Unicode y la codificación de caracteres de idioma” en la página 190

Tareas relacionadas
“Asignación de valores a elementos de datos de grupo (MOVE)” en la página 29
“Conversión a o desde representación nacional (Unicode)” en la página 199

Referencias relacionadas
Clases y categorías de datos (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia MOVE (*COBOL for Linux en x86 Consulta de lenguaje*)

Asignación de valores a elementos de datos de grupo (MOVE)

Utilice la sentencia MOVE para asignar valores a elementos de datos de grupo.

Acerca de esta tarea

Puede mover un elemento de grupo nacional (un elemento de datos que se describe con la cláusula GROUP-USAGE NATIONAL) a otro elemento de grupo nacional. El compilador procesa el movimiento como si cada elemento de grupo nacional fuera un elemento elemental de categoría nacional, es decir, como si cada elemento se describiera como PIC N(*m*), donde *m* es la longitud de ese elemento en posiciones de caracteres nacionales.

Puede mover un elemento de grupo alfanumérico a un elemento de grupo alfanumérico o a un elemento de grupo nacional. También puede mover un elemento de grupo nacional a un elemento de grupo alfanumérico. El compilador realiza estos movimientos como movimientos de grupo, es decir, sin tener en cuenta los elementos elementales individuales en el grupo de envío o de recepción, y sin la conversión del elemento de datos de envío. Asegúrese de que las descripciones de datos subordinadas en los elementos de grupo de envío y recepción son compatibles. Los movimientos se producen incluso si se puede producir un solapamiento destructivo en tiempo de ejecución.

Puede codificar la frase CORRESPONDING en una sentencia MOVE para mover elementos elementales subordinados de un elemento de grupo a los elementos elementales subordinados correspondientes con el mismo nombre en otro elemento de grupo:

```
01  Group-X.
    02 T-Code    Pic X    Value "A".
    02 Month     Pic 99   Value 04.
    02 State     Pic XX   Value "CA".
    02 Filler    PIC X.
01  Group-N     Group-Usage National.
    02 State     Pic NN.
    02 Month     Pic 99.
    02 Filler    Pic N.
    02 Total     Pic 999.
. . .
    MOVE CORR Group-X TO Group-N
```

En el ejemplo anterior, State y Month dentro de Group-N reciben los valores en representación nacional de State y Month, respectivamente, de Group-X. The other data items in Group-N are unchanged. (Filler items in a receiving group item are unchanged by a MOVE CORRESPONDING statement.)

En una sentencia MOVE CORRESPONDING , los elementos de grupo de envío y recepción se tratan como elementos de grupo, no como elementos de datos elementales; se aplica la semántica de grupo. Es decir, se reconocen los elementos de datos elementales dentro de cada grupo y los resultados son los mismos que si se hiciera referencia a cada par de elementos de datos correspondientes en una sentencia MOVE independiente. Las conversiones de datos se realizan de acuerdo con las reglas para la sentencia MOVE tal como se especifica en la referencia relacionada a continuación. Para obtener detalles sobre qué tipos de elementos de datos elementales corresponden, consulte la referencia relacionada sobre la frase CORRESPONDING .

Conceptos relacionados
“Unicode y la codificación de caracteres de idioma” en la página 190
“Grupos nacionales” en la página 198

Tareas relacionadas
“Asignación de valores a elementos de datos elementales (MOVE)” en la página 28
“Utilización de grupos nacionales” en la página 202
“Conversión a o desde representación nacional (Unicode)” en la página 199

Referencias relacionadas
Clases y categorías de elementos de grupo (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia MOVE (*COBOL for Linux en x86 Consulta de lenguaje*)
Frase CORRESPONDIENTE (*COBOL for Linux en x86 Consulta de lenguaje*)

Asignación de resultados aritméticos (MOVE o COMPUTE)

Al asignar un número a un elemento de datos, considere la posibilidad de utilizar la sentencia COMPUTE en lugar de la sentencia MOVE .

Acerca de esta tarea

```
Move w to z
Compute z = w
```

En el ejemplo anterior, las dos sentencias en la mayoría de los casos tienen el mismo efecto. Sin embargo, la sentencia MOVE realiza la asignación con truncamiento. Puede utilizar la opción de compilador DIAGTRUNC para solicitar que el compilador emita un aviso para sentencias MOVE que podrían truncar receptores numéricos.

Cuando se perderían dígitos de orden izquierdo significativos en la ejecución, la sentencia COMPUTE puede detectar la condición y permitirle manejarla. Si utiliza la frase ON SIZE ERROR de la sentencia COMPUTE , el compilador genera código para detectar una condición de desbordamiento de tamaño. Si se produce la condición, el código de la frase ON SIZE ERROR se realiza y el contenido de z permanece sin cambios. Si no especifica la frase ON SIZE ERROR , la asignación se realiza con truncamiento. No hay soporte de ON SIZE ERROR para la sentencia MOVE .

También puede utilizar la sentencia COMPUTE para asignar el resultado de una expresión aritmética o función intrínseca a un elemento de datos. Por ejemplo:

```
Compute z = y + (x ** 3)
Compute x = Function Max(x y z)
```

Referencias relacionadas
“DIAGTRUNC” en la página 280
Funciones intrínsecas (*COBOL for Linux en x86 Consulta de lenguaje*)

Asignación de entrada desde una pantalla o archivo (ACCEPT)

Una forma de asignar un valor a un elemento de datos es leer el valor de una pantalla o un archivo.

Acerca de esta tarea

Para especificar datos desde la pantalla, primero asocie el supervisor con un nombre nemotécnico en el párrafo SPECIAL-NAMES . A continuación, utilice ACCEPT para asignar la línea de entrada especificada en la pantalla a un elemento de datos. Por ejemplo:

```
Environment Division.
Configuration Section.
Special-Names.
    Console is Names-Input.
. . .
    Accept Customer-Name From Names-Input
```

Para leer de un archivo en lugar de la pantalla, realice cualquiera de los cambios siguientes:

- Cambie Console por *dispositivo*, donde *dispositivo* es cualquier dispositivo de sistema válido (por ejemplo, SYSIN). Por ejemplo:

  ```
  SYSIN is Names-Input
  ```

- Establezca la variable de entorno CONSOLE en una especificación de archivo válida utilizando el mandato export . Por ejemplo:

  ```
  export CONSOLE=/myfiles/myinput.rpt
  ```

  El nombre de la variable de entorno debe ser el mismo que el nombre de dispositivo del sistema utilizado. En el ejemplo anterior, el dispositivo del sistema es Console, pero el método de asignar una variable de entorno al nombre de dispositivo del sistema está soportado para todos los dispositivos del

sistema válidos. Por ejemplo, si el dispositivo del sistema es SYSIN, la variable de entorno a la que se debe asignar una especificación de archivo también es SYSIN.

La sentencia ACCEPT asigna la línea de entrada al elemento de datos. Si la línea de entrada es más corta que el elemento de datos, el elemento de datos se rellena con espacios de la representación adecuada. Cuando lee desde una pantalla y la línea de entrada es más larga que el elemento de datos, se descartan los caracteres restantes. Cuando lee de un archivo y la línea de entrada es más larga que el elemento de datos, los caracteres restantes se conservan como la siguiente línea de entrada para el archivo.

Cuando utilice la sentencia ACCEPT , puede asignar un valor a un elemento de grupo nacional o alfanumérico, o a un elemento de datos elemental que tenga USAGE DISPLAY, USAGE DISPLAY-1o USAGE NATIONAL.

Cuando se asigna un valor a un elemento de datos USAGE NATIONAL , los datos de entrada se convierten de la página de códigos asociada con el entorno local de tiempo de ejecución actual a la representación nacional (Unicode UTF-16) sólo si la entrada es del terminal.

Para realizar la conversión cuando los datos de entrada proceden de cualquier otro dispositivo, utilice la función intrínseca NATIONAL-OF .

Conceptos relacionados
“Unicode y la codificación de caracteres de idioma” en la página 190

Tareas relacionadas
“Establecimiento de variables de entorno” en la página 229
“Conversión de alfanuméricos o DBCS a nacional (NATIONAL-OF)” en la página 200
“Obtención de la fecha del sistema en CICS” en la página 410

Referencias relacionadas
sentencia ACCEPT (*COBOL for Linux en x86 Consulta de lenguaje*)
párrafo SPECIAL-NAMES (*COBOL for Linux en x86 Consulta de lenguaje*)

Visualización de valores en una pantalla o en un archivo (DISPLAY)

Puede visualizar el valor de un elemento de datos en una pantalla o grabarlo en un archivo utilizando la sentencia DISPLAY .

Acerca de esta tarea

```
Display "No entry for surname '" Customer-Name "' found in the file.".
```

En el ejemplo anterior, si el contenido del elemento de datos *Customer-Name* es JOHNSON, la sentencia muestra el siguiente mensaje en la pantalla:

```
No entry for surname 'JOHNSON' found in the file.
```

Para grabar datos en un destino que no sea el dispositivo de salida lógica del sistema pantalla, utilice la frase UPON. Por ejemplo, la sentencia siguiente escribe en el archivo que especifique como valor de la variable de entorno SYSOUT:

```
Display "Hello" upon sysout.
```

Cuando visualiza el valor de un elemento de datos USAGE NATIONAL , los datos de salida se convierten a la página de códigos asociada con el entorno local actual.

Conceptos relacionados
“Unicode y la codificación de caracteres de idioma” en la página 190

Tareas relacionadas
“Conversión de nacional a alfanumérico (DISPLAY-OF)” en la página 200
“Codificación de programas COBOL para ejecutarlos en CICS” en la página 409

Referencias relacionadas
“Variables de entorno de ejecución” en la página 234
Sentencia DISPLAY (*COBOL for Linux en x86 Consulta de lenguaje*)

Utilización de funciones intrínsecas (funciones incorporadas)

Algunos lenguajes de programación de alto nivel tienen funciones incorporadas a las que puede hacer referencia en el programa como si fueran variables que tienen atributos definidos y un valor predeterminado. En COBOL, estas funciones se denominan *funciones intrínsecas*. Proporcionan prestaciones para manipular series y números.

Acerca de esta tarea

Puesto que el valor de una función intrínseca se deriva automáticamente en el momento de la referencia, no es necesario definir funciones en DATA DIVISION. Defina sólo los elementos de datos no literales que utilice como argumentos. Las constantes figurativas no están permitidas como argumentos.

Un *identificador-función* es la combinación de la palabra reservada COBOL FUNCTION seguida de un nombre de función (por ejemplo, Max), seguido de cualquier argumento que se vaya a utilizar en la evaluación de la función (por ejemplo, x, y, z). (Opcionalmente, la palabra reservada FUNCTION se puede omitir si se hace referencia al nombre de función en el párrafo REPOSITORY .) Por ejemplo, los grupos de palabras resaltadas a continuación son identificadores de función:

```
Unstring Function Upper-case(Name) Delimited By Space
    Into Fname Lname
Compute A = 1 + Function Log10(x)
Compute M = Function Max(x y z)
```

Un identificador de función representa tanto la invocación de la función como el valor de datos devuelto por la función. Puesto que representa realmente un elemento de datos, puede utilizar un identificador de función en la mayoría de los lugares de PROCEDURE DIVISION donde se puede utilizar un elemento de datos que tiene los atributos del valor devuelto.

La palabra COBOL function es una palabra reservada, pero los nombres de función no están reservados. Puede utilizarlos en otros contextos, como por ejemplo para el nombre de un elemento de datos. Por ejemplo, puede utilizar Sqrt para invocar una función intrínseca y nombrar un elemento de datos en el programa:

```
Working-Storage Section.
01  x                 Pic 99  value 2.
01  y                 Pic 99  value 4.
01  z                 Pic 99  value 0.
01  Sqrt              Pic 99  value 0.
. . .
    Compute Sqrt = 16 ** .5
    Compute z = x + Function Sqrt(y)
    . . .
```

Un identificador de función representa un valor que es de uno de estos tipos: alfanumérico, nacional, numérico o entero. Puede incluir una especificación de subserie (modificador de referencia) en un identificador de función para funciones alfanuméricas o nacionales. Las funciones intrínsecas numéricas se clasifican además de acuerdo con el tipo de números que devuelven.

Las funciones MAX, MIN, DATEVAL, y UNDATE pueden devolver cualquier tipo de valor en función del tipo de argumentos que proporcione.

Las funciones DATEVAL, UNDATEy YEARWINDOW se proporcionan con las extensiones de lenguaje millennium para ayudar a manipular y convertir campos de fecha con ventanas.

Las funciones pueden hacer referencia a otras funciones como argumentos siempre que los resultados de las funciones anidadas cumplan los requisitos para los argumentos de la función externa. Por ejemplo, `Function Sqrt(5)` devuelve un valor numérico. Por lo tanto, los tres argumentos para la función MAX siguiente son todos numéricos, que es un tipo de argumento permitido para esta función:

```
Compute x = Function Max((Function Sqrt(5)) 2.5 3.5)
```

Tareas relacionadas
“Proceso de elementos de tabla utilizando funciones intrínsecas” en la página 80
“Conversión de elementos de datos (funciones intrínsecas)” en la página 106
“Evaluación de elementos de datos (funciones intrínsecas)” en la página 109

Utilización de tablas (matrices) y punteros

En COBOL, las matrices se denominan *tablas*. Una tabla es un conjunto de elementos de datos lógicamente consecutivos que se definen en `DATA DIVISION` utilizando la cláusula `OCCURS` .

Acerca de esta tarea

Los punteros son elementos de datos que contienen direcciones de almacenamiento virtual. Los define explícitamente con la cláusula `USAGE IS POINTER` en `DATA DIVISION` o implícitamente como registros especiales de `ADDRESS OF` .

Puede realizar las siguientes operaciones con elementos de datos de puntero:

- Páselos entre programas utilizando la sentencia `CALL . . . BY REFERENCE` .
- Muévalos a otros punteros utilizando la sentencia `SET` .
- Compárelos con otros punteros para la igualdad utilizando una condición de relación.
- Inicialícelos para que contengan una dirección no válida utilizando `VALUE IS NULL`.

Utilice elementos de datos de puntero para:

- Realice un direccionamiento base limitado, especialmente si desea pasar y recibir direcciones de un área de registro que está definida con `OCCURS DEPENDING ON` y, por lo tanto, está ubicada de forma variable.
- Manejar una lista encadenada.

Tareas relacionadas
“Definición de una tabla (OCCURS)” en la página 59
“Utilización de punteros de procedimiento y función” en la página 481

Capítulo 3. Cómo trabajar con números y aritmética

Acerca de esta tarea

En general, puede ver datos numéricos COBOL como una serie de posiciones de dígitos decimales. Sin embargo, los elementos numéricos también pueden tener propiedades especiales como un signo aritmético o un signo de moneda.

Para definir, visualizar y almacenar datos numéricos de forma que pueda realizar operaciones aritméticas de forma eficiente:

- Utilice la cláusula PICTURE y los caracteres 9, +, -, P, Sy V para definir datos numéricos.
- Utilice la cláusula PICTURE y los caracteres de edición (como Z, coma y punto) junto con las sentencias MOVE y DISPLAY para visualizar datos numéricos.
- Utilice la cláusula USAGE con varios formatos para controlar cómo se almacenan los datos numéricos.
- Utilice la prueba de clase numérica para validar que los valores de datos son adecuados.
- Utilice las sentencias ADD, SUBTRACT, MULTIPLY, DIVIDEy COMPUTE para realizar la aritmética.
- Utilice la cláusula CURRENCY SIGN y los caracteres PICTURE adecuados para designar la moneda que desea.

Tareas relacionadas
“Definición de datos numéricos” en la página 35
“Visualización de datos numéricos” en la página 37
“Control de cómo se almacenan los datos numéricos” en la página 38
“Comprobación de datos incompatibles (prueba de clase numérica)” en la página 48
“Realización aritmética” en la página 49
“Utilización de signos de moneda” en la página 56

Definición de datos numéricos

Defina elementos numéricos utilizando la cláusula PICTURE con el carácter 9 en la descripción de datos para representar los dígitos decimales del número. No utilice un X, que es para elementos de datos alfanuméricos.

Acerca de esta tarea

Por ejemplo, Count-y a continuación es un elemento de datos numérico, un elemento decimal externo que tiene USAGE DISPLAY (un *elemento decimal con zona*):

```
05  Count-y        Pic 9(4)  Value 25.
05  Customer-name  Pic X(20) Value "Johnson".
```

De forma similar, puede definir elementos de datos numéricos para contener caracteres nacionales (UTF-16). Por ejemplo, Count-n a continuación es un elemento de datos decimal externo que tiene USAGE NATIONAL (un *elemento decimal nacional*):

```
05  Count-n        Pic 9(4)  Value 25  Usage National.
```

Puede codificar hasta 18 dígitos en la cláusula PICTURE al compilar utilizando la opción de compilador por omisión ARITH(COMPAT) (denominada *modalidad de compatibilidad*). Cuando compila utilizando ARITH(EXTEND) (al que se hace referencia como *modalidad ampliada*), puede codificar hasta 31 dígitos en la cláusula PICTURE .

Otros caracteres de especial significación que puede codificar son:

P
Indica ceros iniciales o finales

S
Indica un signo, positivo o negativo

V
Implica una coma decimal

s en el ejemplo siguiente significa que el valor está firmado:

```
05  Price  Pic s99v99.
```

Por lo tanto, el campo puede contener un valor positivo o negativo. v indica la posición de una coma decimal implícita, pero no contribuye al tamaño del elemento porque no requiere una posición de almacenamiento. Un s normalmente no contribuye al tamaño de un elemento numérico, porque de forma predeterminada s no requiere una posición de almacenamiento.

Sin embargo, si tiene previsto portar el programa o los datos a una máquina diferente, es posible que desee codificar el signo para un elemento de datos decimal con zona como una posición separada en el almacenamiento. En el caso siguiente, el signo toma 1 byte:

```
05  Price  Pic s99V99  Sign Is Leading, Separate.
```

Esta codificación garantiza que el convenio que la máquina utiliza para almacenar un signo no separado no provocará resultados inesperados en una máquina que utilice un convenio diferente.

Los signos separados también son preferibles para los elementos de datos decimales con zona que se imprimirán o visualizarán.

Se necesitan signos separados para los elementos de datos decimales nacionales que están firmados. El signo toma 2 bytes de almacenamiento, como en el ejemplo siguiente:

```
05 Price Pic s99V99 Usage National Sign Is Leading, Separate.
```

No puede utilizar la cláusula PICTURE con datos de coma flotante internos (COMP-1 o COMP-2). Sin embargo, puede utilizar la cláusula VALUE para proporcionar un valor inicial para un literal de coma flotante interno:

```
05  Compute-result  Usage Comp-2  Value 06.23E-24.
```

Para obtener información sobre los datos de coma flotante externos, consulte los ejemplos a los que se hace referencia a continuación y el concepto relacionado sobre formatos para datos numéricos.

“Ejemplos: datos numéricos y representación interna” en la página 42

Conceptos relacionados
“Formatos para datos numéricos” en la página 39
Apéndice B, “Resultados intermedios y precisión aritmética”, en la página 551

Tareas relacionadas
“Visualización de datos numéricos” en la página 37
“Control de cómo se almacenan los datos numéricos” en la página 38
“Realización aritmética” en la página 49
“Definición de elementos de datos numéricos nacionales” en la página 197

Referencias relacionadas
“Representación de signo de datos con zona y decimal empaquetado” en la página 47
“Almacenamiento de datos de caracteres” en la página 205

“HARIT” en la página 270
Cláusula SIGN (*COBOL for Linux en x86 Consulta de lenguaje*)

Visualización de datos numéricos

Puede definir elementos numéricos con determinados símbolos de edición (como puntos decimales, comas, signos de dólar y signos de débito o crédito) para que los elementos sean más fáciles de leer y comprender al visualizarlos o imprimirlos.

Acerca de esta tarea

Por ejemplo, en el código siguiente, `Edited-price` es un elemento editado numérica-editado que tiene `USAGE DISPLAY`. (Puede especificar la cláusula `USAGE IS DISPLAY` para elementos editados numéricos; sin embargo, está implícita. Significa que los elementos se almacenan en formato de caracteres.)

```
05  Price          Pic     9(5)v99.
05  Edited-price   Pic  $zz,zz9.99.
. . .
Move Price To Edited-price
Display Edited-price
```

Si el contenido de `Price` es 0150099 (que representa el valor 1.500,99), se visualiza `$ 1,500.99` al ejecutar el código. `z` en la cláusula `PICTURE` de `Edited-price` indica la supresión de ceros iniciales.

Puede definir elementos de datos editados numéricos para contener caracteres nacionales (UTF-16) en lugar de caracteres alfanuméricos. Para ello, defina los elementos editados numéricos como `USAGE NATIONAL`. El efecto de los símbolos de edición es el mismo para los elementos editados numéricos que tienen `USAGE NATIONAL` que para los elementos editados numéricos que tienen `USAGE DISPLAY`, excepto que la edición se realiza con caracteres nacionales. Por ejemplo, si `Edited-price` se declara como `USAGE NATIONAL` en el código anterior, el elemento se edita y se visualiza utilizando caracteres nacionales.

Puede hacer que un elemento numérico elemental o editado numérica-editado se rellene con espacios cuando se almacene un valor de cero en él codificando la cláusula `BLANK WHEN ZERO` para el elemento. Por ejemplo, cada una de las sentencias `DISPLAY` siguientes hace que se muestren espacios en blanco en lugar de ceros:

```
05 Price           Pic     9(5)v99.
05 Edited-price-D  Pic  $99,999.99
       Blank When Zero.
05 Edited-price-N  Pic  $99,999.99 Usage National
       Blank When Zero.
. . .
Move 0 to Price
Move Price to Edited-price-D
Move Price to Edited-price-N
Display Edited-price-D
Display Edited-price-N
```

No puede utilizar elementos editados numéricos como envío de operandos en expresiones aritméticas o en sentencias `ADD`, `SUBTRACT`, `MULTIPLY`, `DIVIDE`o `COMPUTE` . (La edición numérica tiene lugar cuando un elemento editado numéricamente es el campo de recepción de una de estas sentencias, o cuando una sentencia `MOVE` tiene un campo de recepción editado numéricamente y un campo de envío numérico o editado numéricamente.) Los elementos editados numéricos se utilizan principalmente para visualizar o imprimir datos numéricos.

Puede mover elementos editados numéricos a elementos numéricos o editados numéricos. En el ejemplo siguiente, el valor del elemento editado numérica-editado (si tiene USAGE DISPLAY o USAGE NATIONAL) se mueve al elemento numérico:

```
Move Edited-price to Price
Display Price
```

Si estas dos sentencias siguen inmediatamente a las sentencias del primer ejemplo anterior, Price se mostrará como 0150099, que representa el valor 1.500,99. Price también se visualizaría como 0150099 si Edited-price tuviera USAGE NATIONAL.

También puede mover elementos editados numéricos a elementos de datos alfanuméricos, editados alfanuméricos, de coma flotante y nacionales. Para obtener una lista completa de los elementos de recepción válidos para datos editados numéricos, consulte la referencia relacionada sobre la sentencia MOVE .

"Ejemplos: datos numéricos y representación interna" en la página 42

Tareas relacionadas

"Visualización de valores en una pantalla o en un archivo (DISPLAY)" en la página 32
"Control de cómo se almacenan los datos numéricos" en la página 38
"Definición de datos numéricos" en la página 35
"Realización aritmética" en la página 49
"Definición de elementos de datos numéricos nacionales" en la página 197
"Conversión a o desde representación nacional (Unicode)" en la página 199

Referencias relacionadas

sentencia MOVE (*COBOL for Linux en x86 Consulta de lenguaje*)
cláusula BLANK WHEN ZERO (*COBOL for Linux en x86 Consulta de lenguaje*)

Control de cómo se almacenan los datos numéricos

Puede controlar cómo el sistema almacena los elementos de datos numéricos codificando la cláusula USAGE en las entradas de descripción de datos.

Acerca de esta tarea

Es posible que desee controlar el formato por varias razones como las siguientes:

- La aritmética realizada con tipos de datos computacionales es más eficiente que con los tipos de datos USAGE DISPLAY o USAGE NATIONAL .
- El formato decimal empaquetado requiere menos almacenamiento por dígito que los tipos de datos USAGE DISPLAY o USAGE NATIONAL .
- El formato decimal empaquetado se convierte a y desde el formato DISPLAY o NATIONAL de forma más eficiente que el formato binario.
- El formato de coma flotante es adecuado para operandos aritméticos y resultados con una escala muy variable, manteniendo al mismo tiempo el número máximo de dígitos significativos.
- Es posible que tenga que conservar los formatos de datos al mover datos de una máquina a otra.

Los datos numéricos que utilice en el programa tendrán uno de los siguientes formatos disponibles con COBOL:

- Decimal externo (USAGE DISPLAY o USAGE NATIONAL)
- Coma flotante externa (USAGE DISPLAY o USAGE NATIONAL)
- Decimal interno (USAGE PACKED-DECIMAL)
- Binario (USAGE BINARY)
- Binario nativo (USAGE COMP-5)

- Coma flotante interna (USAGE COMP-1 o USAGE COMP-2)

COMP y COMP-4 son sinónimos de BINARYy COMP-3 es sinónimo de PACKED-DECIMAL.

El compilador convierte los números visualizables en la representación interna de sus valores numéricos antes de utilizarlos en operaciones aritméticas. Por lo tanto, a menudo es más eficiente si define elementos de datos como BINARY o PACKED-DECIMAL que como DISPLAY o NATIONAL. Por ejemplo:

```
05  Initial-count   Pic S9(4)  Usage Binary   Value 1000.
```

Independientemente de la cláusula USAGE que utilice para controlar la representación interna de un valor, utilice los mismos convenios de cláusula PICTURE y el mismo valor decimal en la cláusula VALUE (excepto para los datos de coma flotante internos , para el que no puede utilizar una cláusula PICTURE).

"Ejemplos: datos numéricos y representación interna" en la página 42

Conceptos relacionados
"Formatos para datos numéricos" en la página 39
"Conversiones de formato de datos" en la página 46
Apéndice B, "Resultados intermedios y precisión aritmética", en la página 551

Tareas relacionadas
"Definición de datos numéricos" en la página 35
"Visualización de datos numéricos" en la página 37
"Realización aritmética" en la página 49

Referencias relacionadas
"Conversiones y precisión" en la página 46
"Representación de signo de datos con zona y decimal empaquetado" en la página 47

Formatos para datos numéricos

Hay varios formatos disponibles para los datos numéricos.

Elementos decimales externos (DISPLAY y NATIONAL)

Cuando USAGE DISPLAY está en vigor para un elemento de datos numérico de categoría (ya sea porque lo ha codificado, o de forma predeterminada), cada posición (byte) de almacenamiento contiene un dígito decimal. Los elementos se almacenan en forma visualizable. Los elementos decimales externos que tienen USAGE DISPLAY se conocen como elementos de datos *decimales con zona* .

Cuando USAGE NATIONAL está en vigor para un elemento de datos numérico de categoría, se necesitan 2 bytes de almacenamiento para cada dígito decimal. Los elementos se almacenan en formato UTF-16 . Los elementos decimales externos que tienen USAGE NATIONAL se conocen como elementos de datos *decimales nacionales* .

Los elementos de datos decimales nacionales, si están firmados, deben tener la cláusula SIGN SEPARATE en vigor. Todas las demás reglas para los elementos decimales con zona se aplican a los elementos decimales nacionales. Puede utilizar elementos decimales nacionales en cualquier lugar en el que se puedan utilizar otros elementos de datos numéricos de categoría.

Los elementos de datos decimales externos (decimales con zona y decimales nacionales) están pensados principalmente para recibir y enviar números entre el programa y los archivos, terminales o impresoras. También puede utilizar elementos decimales externos como operandos y receptores en el proceso aritmético. Sin embargo, si el programa realiza una gran cantidad de aritmética intensiva y la eficiencia es una prioridad alta, los tipos numéricos computacionales de COBOL podrían ser una mejor opción para los elementos de datos utilizados en la aritmética.

Elementos externos de coma flotante (DISPLAY y NATIONAL)

Cuando USAGE DISPLAY está en vigor para un elemento de datos de coma flotante (ya sea porque lo ha codificado, o de forma predeterminada), cada posición de carácter PICTURE (excepto para v, una coma decimal implícita, si se utiliza) toma 1 byte de almacenamiento. Los elementos se almacenan en forma visualizable. Los elementos de coma flotante externos que tienen USAGE DISPLAY se conocen como *mostrar elementos de datos de coma flotante* en esta información cuando sea necesario para distinguirlos de los elementos de coma flotante externos que tienen USAGE NATIONAL.

En el ejemplo siguiente, Compute-Result se define implícitamente como un elemento de coma flotante display :

```
05  Compute-Result  Pic -9v9(9)E-99.
```

Los signos menos (-) no significan que la mantisa y el exponente deben ser necesariamente números negativos. En su lugar, significan que cuando se visualiza el número, el signo aparece como un espacio en blanco para los números positivos o un signo menos para los números negativos. Si en su lugar codifica un signo más (+), el signo aparece como un signo más para los números positivos o un signo menos para los números negativos.

Cuando USAGE NATIONAL está en vigor para un elemento de datos de coma flotante, cada posición de carácter PICTURE (excepto para v, si se utiliza) toma 2 bytes de almacenamiento. Los elementos se almacenan como caracteres nacionales (UTF-16). Los elementos de coma flotante externos que tienen USAGE NATIONAL se conocen como elementos de datos de *coma flotante nacional* .

Las reglas existentes para visualizar elementos de coma flotante se aplican a los elementos de coma flotante nacionales.

En el ejemplo siguiente, Compute-Result-N es un elemento de coma flotante nacional:

```
05  Compute-Result-N  Pic -9v9(9)E-99  Usage National.
```

Si se visualiza Compute-Result-N , los signos aparecen como se ha descrito anteriormente para Compute-Result, pero en caracteres nacionales.

No puede utilizar la cláusula VALUE para elementos de coma flotante externos.

Al igual que con los números decimales externos, los números de coma flotante externos deben ser convertidos (por el compilador) a una representación interna de su valor numérico antes de que puedan ser utilizados en operaciones aritméticas. Si compila con la opción predeterminada ARITH (COMPAT), los números de coma flotante externos se convierten al formato de coma flotante largo (64 bits). Si compila con ARITH (EXTEND), en su lugar se convierten al formato de coma flotante (128 bits) de precisión ampliada.

Elementos binarios (COMP)

BINARY, COMPy COMP-4 son sinónimos. Los números de formato binario ocupan 2, 4 u 8 bytes de almacenamiento. Si la cláusula PICTURE especifica que un elemento está firmado, el bit situado más a la izquierda se utiliza como signo operativo.

Un número binario con una descripción PICTURE de cuatro o menos dígitos decimales ocupa 2 bytes; de cinco a nueve dígitos decimales, 4 bytes; y de 10 a 18 dígitos decimales, 8 bytes. Los elementos binarios con nueve o más dígitos requieren más manejo por parte del compilador.

Puede utilizar elementos binarios, por ejemplo, para índices, subíndices, conmutadores y operandos aritméticos o resultados.

Utilice la opción de compilador TRUNC(STD|OPT|BIN) para indicar cómo se truncarán los datos binarios (BINARY, COMPo COMP-4).

Elementos binarios nativos (COMP-5)

Los elementos de datos que define como USAGE COMP-5 se representan en el almacenamiento como datos binarios. Sin embargo, a diferencia de los elementos USAGE COMP , pueden contener valores de magnitud hasta la capacidad de la representación binaria nativa (2, 4 u 8 bytes) en lugar de limitarse al valor implícito por el número de 9en la cláusula PICTURE .

Cuando mueve o almacena datos numéricos en un elemento COMP-5 , el truncamiento se produce en el tamaño del campo binario en lugar de en el límite de tamaño de COBOL PICTURE . Cuando se hace referencia a un elemento COMP-5 , se utiliza el tamaño de campo binario completo en la operación.

Por lo tanto, COMP-5 es especialmente útil para elementos de datos binarios que se originan en programas no COBOL en los que los datos podrían no ajustarse a una cláusula PICTURE de COBOL.

La tabla siguiente muestra los rangos de valores posibles para los elementos de datos de COMP-5 .

Tabla 3. ***Rangos en valor de elementos de datos COMP-5***

PICTURE	**Representación de almacenamiento**	**Valores numéricos**
S9(1) a S9(4)	Media palabra binaria (2 bytes)	-32768 a +32767
S9(5) a S9(9)	Palabra completa binaria (4 bytes)	-2.147.483.648 a +2.147.483.647
S9(10) a S9(18)	Palabra doble binaria (8 bytes)	-9.223.372.036.854.775.808 a +9.223.372.036.854.775.807
9(1) a 9(4)	Media palabra binaria (2 bytes)	0 a 65535
9(5) a 9(9)	Palabra completa binaria (4 bytes)	0 a 4.294.967.295
9(10) a 9(18)	Palabra doble binaria (8 bytes)	0 a 18.446.744.073.709.551.615

Puede especificar el escalado (es decir, posiciones decimales o posiciones enteras implícitas) en la cláusula PICTURE de los elementos COMP-5 . Si lo hace, debe escalar adecuadamente las capacidades máximas enumeradas anteriormente. Por ejemplo, un elemento de datos que describe como PICTURE S99V99 COMP-5 se representa en el almacenamiento como una media palabra binaria y soporta un rango de valores de -327.68 a +327.67.

Literales grandes en cláusulas VALUE : los literales especificados en cláusulas VALUE para elementos COMP-5 pueden, con algunas excepciones, contener valores de magnitud hasta la capacidad de la representación binaria nativa. Consulte *COBOL for Linux en x86 Consulta de lenguaje* para ver las excepciones.

Independientemente del valor de la opción de compilador TRUNC , los elementos de datos COMP-5 se comportan como lo hacen los datos binarios en los programas compilados con TRUNC(BIN).

Elementos de decimal empaquetado (COMP-3)

PACKED-DECIMAL y COMP-3 son sinónimos. Los elementos de decimal empaquetado ocupan 1 byte de almacenamiento para cada dos dígitos decimales que codifique en la descripción PICTURE , excepto que el byte más a la derecha contiene sólo un dígito y el signo. Este formato es más eficaz cuando se codifica un número impar de dígitos en la descripción PICTURE , de modo que se utiliza completamente el byte situado más a la izquierda. Los elementos decimales empaquetados se manejan como números de coma fija para fines aritméticos.

Elementos internos de coma flotante (COMP-1 y COMP-2)

COMP-1 hace referencia al formato de coma flotante corto y COMP-2 hace referencia al formato de coma flotante largo, que ocupa 4 y 8 bytes de almacenamiento, respectivamente.

Los elementos de datos COMP-1 y COMP-2 se representan en formato IEEE si la opción de compilador FLOAT(NATIVE) (el valor predeterminado) está en vigor. Si FLOAT(BE) está en vigor, los elementos de datos COMP-1 y COMP-2 se representan de forma coherente con System z, es decir, en formato hexadecimal de coma flotante. Para obtener detalles, consulte la referencia relacionada sobre la opción FLOAT .

Conceptos relacionados
“Unicode y la codificación de caracteres de idioma” en la página 190
Apéndice B, “Resultados intermedios y precisión aritmética”, en la página 551

Tareas relacionadas
“Definición de datos numéricos” en la página 35
“Definición de elementos de datos numéricos nacionales” en la página 197

Referencias relacionadas
“Almacenamiento de datos de caracteres” en la página 205
“TRUNC” en la página 302
“FLOAT” en la página 286
Clases y categorías de datos (*COBOL for Linux en x86 Consulta de lenguaje*)
Cláusula SIGN (*COBOL for Linux en x86 Consulta de lenguaje*)
Cláusula VALUE (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplos: datos numéricos y representación interna

La tabla que muestran la representación interna de elementos numéricos.

La tabla siguiente muestra la representación interna de elementos numéricos para tipos de datos binarios.

Tabla 4. ***Representación interna de elementos numéricos binarios***

Tipo numérico	Cláusula PICTURE y USAGE y opcional SIGN	Valor	Representación interna
Binario	PIC S9999 BINARY PIC S9999 COMP PIC S9999 COMP-4	+ 1234	D2 04
		- 1234	2E FB
	PIC S9999 COMP-5	+ 12345[1]	39 30
		- 12345[1]	C7 CF
	PIC 9999 BINARY PIC 9999 COMP PIC 9999 COMP-4	1234	D2 04
	PIC 9999 COMP-5	60000[1]	60 EA

1. El ejemplo demuestra que los elementos de datos COMP-5 pueden contener valores de magnitud hasta la capacidad de la representación binaria nativa (2, 4 u 8 bytes), en lugar de limitarse al valor implícito por el número de 9en la cláusula PICTURE .

La tabla siguiente muestra la representación interna de elementos numéricos en formato de datos nativo. Supongamos que las opciones de compilador `CHAR(NATIVE)` y `FLOAT(NATIVE)` están en vigor.

Tabla 5. ***Representación interna de elementos numéricos nativos***

Tipo numérico	**Cláusula PICTURE y USAGE y opcional SIGN**	**Valor**	**Representación interna**
Decimal externo	`PIC S9999 DISPLAY`	`+ 1234`	`31 32 33 34`
		`- 1234`	`31 32 33 74`
		`  1234`	`31 32 33 34`
	`PIC 9999 DISPLAY`	`  1234`	`31 32 33 34`
	`PIC 9999 NATIONAL`	`  1234`	`31 00 32 00 33 00 34 00`
	`PIC S9999 DISPLAY SIGN LEADING`	`+ 1234`	`31 32 33 34`
		`- 1234`	`71 32 33 34`
	`PIC S9999 DISPLAY SIGN LEADING SEPARATE`	`+ 1234`	`2B 31 32 33 34`
		`- 1234`	`2D 31 32 33 34`
	`PIC S9999 DISPLAY SIGN TRAILING SEPARATE`	`+ 1234`	`31 32 33 34 2B`
		`- 1234`	`31 32 33 34 2D`
	`PIC S9999 NATIONAL SIGN LEADING SEPARATE`	`+ 1234`	`2B 00 31 00 32 00 33 00 34 00`
		`- 1234`	`2D 00 31 00 32 00 33 00 34 00`
	`PIC S9999 NATIONAL SIGN TRAILING SEPARATE`	`+ 1234`	`31 00 32 00 33 00 34 00 2B 00`
		`- 1234`	`31 00 32 00 33 00 34 00 2D 00`
Decimal interno	`PIC S9999 PACKED-DECIMAL` `PIC S9999 COMP-3`	`+ 1234`	`01 23 4C`
		`- 1234`	`01 23 4D`
	`PIC 9999  PACKED-DECIMAL` `PIC 9999  COMP-3`	`  1234`	`01 23 4C`
Coma flotante interna	`COMP-1`	`+ 1234`	`00 40 9A 44`
		`- 1234`	`00 40 9A C4`
	`COMP-2`	`+ 1234`	`00 00 00 00 00 48 93 40`
		`- 1234`	`00 00 00 00 00 48 93 C0`

*Tabla 5. **Representación interna de elementos numéricos nativos** (continuación)*			
Tipo numérico	**Cláusula PICTURE y USAGE y opcional SIGN**	**Valor**	**Representación interna**
Coma flotante externa	PIC +9(2).9(2)E+99 DISPLAY	+ 12.34E+02	2B 31 32 2E 33 34 45 2B 30 32
		- 12.34E+02	2D 31 32 2E 33 34 45 2B 30 32
	PIC +9(2).9(2)E+99 NATIONAL	+ 12.34E+02	2B 00 31 00 32 00 2E 00 33 00 34 00 45 00 2B 00 30 00 32 00
		- 12.34E+02	2D 00 31 00 32 00 2E 00 33 00 34 00 45 00 2B 00 30 00 32 00

La tabla siguiente muestra la representación interna de elementos numéricos en formato de datos de host IBM Z . Supongamos que las opciones de compilador CHAR(EBCDIC) y FLOAT(BE) están en vigor.

Tabla 6. ***Representación interna de elementos numéricos cuando CHAR (EBCDIC) y FLOAT (BE) están en vigor***			
Tipo numérico	**Cláusula PICTURE y USAGE y opcional SIGN**	**Valor**	**Representación interna**
Decimal externo	PIC S9999 DISPLAY	+ 1234	F1 F2 F3 C4
		- 1234	F1 F2 F3 D4
		1234	F1 F2 F3 C4
	PIC 9999 DISPLAY	1234	F1 F2 F3 F4
	PIC 9999 NATIONAL	1234	00 31 00 32 00 33 00 34
	PIC S9999 DISPLAY SIGN LEADING	+ 1234	C1 F2 F3 F4
		- 1234	D1 F2 F3 F4
	PIC S9999 DISPLAY SIGN LEADING SEPARATE	+ 1234	4E F1 F2 F3 F4
		- 1234	60 F1 F2 F3 F4
	PIC S9999 DISPLAY SIGN TRAILING SEPARATE	+ 1234	F1 F2 F3 F4 4E
		- 1234	F1 F2 F3 F4 60
	PIC S9999 NATIONAL SIGN LEADING SEPARATE	+ 1234	00 2B 00 31 00 32 00 33 00 34
		- 1234	00 2D 00 31 00 32 00 33 00 34
	PIC S9999 NATIONAL SIGN TRAILING SEPARATE	+ 1234	00 31 00 32 00 33 00 34 00 2B
		- 1234	00 31 00 32 00 33 00 34 00 2D
Decimal interno	PIC S9999 PACKED-DECIMAL PIC S9999 COMP-3	+ 1234	01 23 4C
		- 1234	01 23 4D
	PIC 9999 PACKED-DECIMAL PIC 9999 COMP-3	1234	01 23 4C
Coma flotante interna	COMP-1	+ 1234	43 4D 20 00
		- 1234	C3 4D 20 00
	COMP-2	+ 1234	43 4D 20 00 00 00 00 00
		- 1234	C3 4D 20 00 00 00 00 00

Tabla 6. ***Representación interna de elementos numéricos cuando CHAR (EBCDIC) y FLOAT (BE) están en vigor*** *(continuación)*

Tipo numérico	Cláusula PICTURE y USAGE y opcional SIGN	Valor	Representación interna
Coma flotante externa	PIC +9(2).9(2)E+99 DISPLAY	+ 12.34E+02	4E F1 F2 4B F3 F4 C5 4E F0 F2
		- 12.34E+02	60 F1 F2 4B F3 F4 C5 4E F0 F2
	PIC +9(2).9(2)E+99 NATIONAL	+ 12.34E+02	00 2B 00 31 00 32 00 2E 00 33 00 34 00 45 00 2B 00 30 00 32
		- 12.34E+02	00 2D 00 31 00 32 00 2E 00 33 00 34 00 45 00 2B 00 30 00 32

Conversiones de formato de datos

Cuando el código del programa implica la interacción de elementos que tienen distintos formatos de datos, el compilador convierte estos elementos temporalmente, para comparaciones y operaciones aritméticas, o de forma permanente, para la asignación al receptor en una MOVE, COMPUTEu otra sentencia aritmética.

Una conversión es en realidad un movimiento de un valor de un elemento de datos a otro. El compilador realiza las conversiones necesarias durante la ejecución de aritméticas o comparaciones utilizando las mismas reglas que se utilizan para las sentencias MOVE y COMPUTE .

Cuando es posible, el compilador realiza un movimiento para conservar el valor numérico en lugar de un movimiento directo dígito por dígito.

La conversión generalmente requiere almacenamiento adicional y tiempo de proceso porque los datos se mueven a un área de trabajo interna y se convierten antes de que se realice la operación. Es posible que también sea necesario volver a mover los resultados a un área de trabajo y convertirlos de nuevo.

Las conversiones entre formatos de datos de coma fija (decimal externo, decimal empaquetado o binario) se realizan sin pérdida de precisión siempre que el campo de destino pueda contener todos los dígitos del operando de origen.

Es posible una pérdida de precisión en las conversiones entre formatos de datos de coma fija y formatos de datos de coma flotante (coma flotante corto, coma flotante largo o coma flotante externo). Estas conversiones se producen durante las evaluaciones aritméticas que tienen una mezcla de operandos de punto fijo y de coma flotante.

Referencias relacionadas
“Conversiones y precisión” en la página 46
“Representación de signo de datos con zona y decimal empaquetado” en la página 47

Conversiones y precisión

En algunas conversiones numéricas, es posible una pérdida de precisión; otras conversiones conservan la precisión o dan como resultado el redondeo.

Puesto que tanto los elementos de coma fija como los de coma flotante externa tienen características decimales, las referencias a los elementos de coma fija en los ejemplos siguientes incluyen elementos de coma flotante externos a menos que se indique lo contrario.

Cuando el compilador convierte de punto fijo a formato de coma flotante interno, los números de punto fijo en base 10 se convierten al sistema de numeración utilizado internamente.

Cuando el compilador convierte el formato corto a formato largo para las comparaciones, se utilizan ceros para rellenar el número más corto.

Conversiones que pierden precisión

Cuando un elemento de datos USAGE COMP-2 se mueve a un elemento de datos de punto fijo que tiene más de 18 dígitos, el elemento de datos de punto fijo recibirá sólo 18 dígitos significativos y los dígitos restantes serán cero.

Cuando un elemento de datos USAGE COMP-1 se mueve a un elemento de datos de punto fijo que tiene más de seis dígitos, el elemento de datos de punto fijo sólo recibirá seis dígitos significativos y los dígitos restantes serán cero.

Conversiones que conservan la precisión

Si un elemento de datos de punto fijo que tiene seis o menos dígitos se mueve a un elemento de datos USAGE COMP-1 y, a continuación, se devuelve al elemento de datos de punto fijo, se recupera el valor original.

Si un elemento de datos USAGE COMP-1 se mueve a un elemento de datos de punto fijo de seis o más dígitos y, a continuación, se devuelve al elemento de datos USAGE COMP-1 , se recupera el valor original.

Si un elemento de datos de punto fijo que tiene 15 dígitos o menos se mueve a un elemento de datos USAGE COMP-2 y, a continuación, se devuelve al elemento de datos de punto fijo, se recupera el valor original.

Si un elemento de datos USAGE COMP-2 se mueve a un elemento de datos de punto fijo (no de coma flotante externa) de 18 o más dígitos y, a continuación, se devuelve al elemento de datos USAGE COMP-2 , el valor original se recupera.

Conversiones que dan como resultado el redondeo

Si un elemento de datos USAGE COMP-1 , un elemento de datos USAGE COMP-2 , un elemento de datos de coma flotante externo o un literal de coma flotante se mueve a un elemento de datos de punto fijo, el redondeo se produce en la posición de orden inferior del elemento de datos de destino. Los valores fraccionales .5 y mayores se redondean hacia arriba; los valores fraccionales menores que .5 se redondean hacia abajo.

Si un elemento de datos USAGE COMP-2 se mueve a un elemento de datos USAGE COMP-1 , el redondeo se produce en la posición de orden inferior del elemento de datos de destino.

Si un elemento de datos de punto fijo se mueve a un elemento de datos de punto flotante externo y el PICTURE del elemento de datos de punto fijo contiene más posiciones de dígito que el PICTURE del elemento de datos de punto flotante externo, el redondeo se produce en la posición de orden inferior del elemento de datos de destino.

Conceptos relacionados

Apéndice B, “Resultados intermedios y precisión aritmética”, en la página 551

Representación de signo de datos con zona y decimal empaquetado

La representación de signo afecta al proceso y a la interacción de los datos decimal con zona y decimal interno .

Dado X'*sd*', donde *s* es la representación de signo y *d* representa el dígito, las representaciones de signo válidas para datos decimales con zona (USAGE DISPLAY) sin la cláusula SIGN IS SEPARATE son:

Positivo:
3, Cy F

Negativo:
7 y D

Cuando la opción de compilador CHAR(NATIVE) está en vigor, los signos generados internamente son 3 para positivos y sin signo, y 7 para negativos.

Cuando la opción de compilador CHAR(EBCDIC) está en vigor, los signos generados internamente son C para positivo, F para no firmado y D para negativo.

Dados X'*ds*', donde *d* representa el dígito y *s* es la representación de signo, las representaciones de signo válidas para los datos decimales internos (USAGE PACKED-DECIMAL) son:

Positivo:
A, C, Ey F

Negativo:
B y D

Los signos generados internamente son C para positivos y no firmados, y D para negativos.

La representación de signo de números decimales internos sin signo es diferente entre COBOL para Linux y Enterprise COBOL for z/OS. Enterprise COBOL for z/OS genera F internamente como signo de números decimales internos sin signo.

Referencias relacionadas
“ZWB” en la página 307
"Representación de datos" en la *Guía de migración*

Comprobación de datos incompatibles (prueba de clase numérica)

El compilador presupone que los valores que proporcione para un elemento de datos son válidos para las cláusulas PICTURE y USAGE , y no comprueba su validez. Asegúrese de que el contenido de un elemento de datos se ajusta a las cláusulas PICTURE y USAGE antes de utilizar el elemento en un proceso adicional.

Acerca de esta tarea

Puede suceder que los valores se pasen al programa y se asignen a elementos que tengan descripciones de datos incompatibles para dichos valores. Por ejemplo, los datos no numéricos se pueden mover o pasar a un campo definido como numérico, o un número con signo se puede pasar a un campo definido como sin signo. En cualquier caso, los campos de recepción contienen datos no válidos. Cuando se proporciona a un elemento un valor que es incompatible con su descripción de datos, las referencias a dicho elemento en PROCEDURE DIVISION no están definidas y los resultados son imprevisibles.

Puede utilizar la prueba de clase numérica para realizar la validación de datos. Por ejemplo:

```
Linkage Section.
01  Count-x   Pic 999.
. . .
Procedure Division Using Count-x.
    If Count-x is numeric then display "Data is good"
```

La prueba de clase numérica comprueba el contenido de un elemento de datos con un conjunto de valores que son válidos para PICTURE y USAGE del elemento de datos.

Realización aritmética

Puede utilizar cualquiera de varias características de lenguaje COBOL (incluyendo COMPUTE, expresiones aritméticas, funciones intrínsecas numéricasy servicios invocables de fecha/hora) para realizar la aritmética. Su elección depende de si una característica satisface sus necesidades particulares.

Acerca de esta tarea

Para las evaluaciones aritméticas más comunes, la sentencia COMPUTE es adecuada. Si necesita utilizar literales numéricos, datos numéricos u operadores aritméticos, es posible que desee utilizar expresiones aritméticas. En los lugares donde se permiten expresiones numéricas, puede ahorrar tiempo utilizando funciones intrínsecas numéricas.

Tareas relacionadas
“Utilización de COMPUTE y otras sentencias aritméticas” en la página 49
“Utilización de expresiones aritméticas” en la página 50
“Utilización de funciones intrínsecas numéricas” en la página 50

Utilización de COMPUTE y otras sentencias aritméticas

Utilice la sentencia COMPUTE para la mayoría de las evaluaciones aritméticas en lugar de las sentencias ADD, SUBTRACT, MULTIPLYy DIVIDE . A menudo sólo puede codificar una sentencia COMPUTE en lugar de varias sentencias aritméticas individuales.

Acerca de esta tarea

La sentencia COMPUTE asigna el resultado de una expresión aritmética a uno o más elementos de datos:

```
Compute z     = a + b / c ** d - e
Compute x y z = a + b / c ** d - e
```

Algunos cálculos aritméticos pueden ser más intuitivos utilizando sentencias aritméticas distintas de COMPUTE. Por ejemplo:

COMPUTE	Sentencias aritméticas equivalentes
`Compute Increment = Increment + 1`	`Add 1 to Increment`
`Compute Balance =` `  Balance - Overdraft`	`Subtract Overdraft from Balance`

`Compute IncrementOne =`
`  IncrementOne + 1`
`Compute IncrementTwo =`
`  IncrementTwo + 1`
`Compute IncrementThree =`
`  IncrementThree + 1`
	`Add 1 to IncrementOne,`
	`  IncrementTwo,`
	`  IncrementThree`

También es posible que prefiera utilizar la sentencia DIVIDE (con su frase REMAINDER) para la división en la que desea procesar un resto. La función intrínseca de REM también proporciona la capacidad de procesar un resto.

Cuando realiza cálculos aritméticos, puede utilizar elementos de datos decimales nacionales como operandos, al igual que utiliza elementos de datos decimales con zona. También puede utilizar elementos

de datos de coma flotante nacionales como operandos al igual que utiliza los operandos de coma flotante de visualización.

Conceptos relacionados
"Punto fijo contrastado con aritmética de coma flotante" en la página 54
Apéndice B, "Resultados intermedios y precisión aritmética", en la página 551

Tareas relacionadas
"Definición de datos numéricos" en la página 35

Utilización de expresiones aritméticas

Puede utilizar expresiones aritméticas en muchos lugares (pero no en todos) de las sentencias donde se permiten elementos de datos numéricos.

Acerca de esta tarea

Por ejemplo, puede utilizar expresiones aritméticas como comparandos en condiciones de relación:

```
If (a + b) > (c - d + 5) Then. . .
```

Las expresiones aritméticas pueden constar de un único literal numérico, un único elemento de datos numéricos o una única referencia de función intrínseca. También pueden consistir en varios de estos elementos conectados por operadores aritméticos.

Los operadores aritméticos se evalúan en el siguiente orden de prioridad:

Tabla 7. ***Orden de evaluación de los operadores aritméticos***

Operador	**Significado**	**Orden de evaluación**
Unary + o-	Signo algebraico	Primera
**	Exponenciación	Segundo
/o *	División o multiplicación	Tercero
Binario + o-	Suma o resta	Último

Los operadores con el mismo nivel de prioridad se evalúan de izquierda a derecha; sin embargo, puede utilizar paréntesis para cambiar el orden de evaluación. Las expresiones entre paréntesis se evalúan antes de que se evalúen los operadores individuales. Los paréntesis, sean necesarios o no, hacen que su programa sea más fácil de leer.

Conceptos relacionados
"Punto fijo contrastado con aritmética de coma flotante" en la página 54
Apéndice B, "Resultados intermedios y precisión aritmética", en la página 551

Utilización de funciones intrínsecas numéricas

Puede utilizar funciones intrínsecas numéricas sólo en lugares en los que se permiten expresiones numéricas. Estas funciones pueden ahorrarle tiempo porque no tiene que codificar los muchos tipos comunes de cálculos que proporcionan.

Acerca de esta tarea

Las funciones intrínsecas numéricas devuelven un valor numérico con signo y se tratan como elementos de datos numéricos temporales.

Las funciones numéricas se clasifican en las categorías siguientes:

Entero
Aquellos que devuelven un entero

coma flotante
Aquellos que devuelven un valor de coma flotante largo (64 bits) o de precisión ampliada (128 bits) (dependiendo de si compila utilizando la opción predeterminada `ARITH(COMPAT)` o utilizando `ARITH(EXTEND)`)

Combinado
Aquellos que devuelven un entero, un valor de coma flotante o un número de coma fija con posiciones decimales, en función de los argumentos

Puede utilizar funciones intrínsecas para realizar varias operaciones aritméticas diferentes, tal como se describe en la tabla siguiente.

Tabla 8. ***Funciones intrínsecas numéricas***

Manejo de números	Fecha y hora	Finanzas	Matemáticas	Estadísticas
LENGTH	ADD-DURATION	ANNUITY	ACOS	MEAN
MAX	CONVERT-DATE-TIME	PRESENT-VALUE	ASIN	MEDIAN
MIN	CURRENT-DATE		ATAN	MIDRANGE
NUMVAL	DATE-OF-INTEGER		COS	RANDOM
NUMVAL-C	DATE-TO-YYYYMMDD		FACTORIAL	RANGE
	DATEVAL		INTEGER	STANDARD-DEVIATION
	DAY-OF-INTEGER		INTEGER-PART	VARIANCE
	DAY-TO-YYYYDDD		LOG	
	EXTRACT-DATE-TIME		LOG10	
	FIND-DURATION		MOD	
ORD-MAX	INTEGER-OF-DATE		REM	
ORD-MIN	INTEGER-OF-DAY		SIN	
	UNDATE		SQRT	
	SUBTRACT-DURATION		SUM	
	TEST-DATE-TIME		TAN	
	WHEN-COMPILED			
	YEAR-TO-YYYY			
	YEARWINDOW			

“Ejemplos: funciones intrínsecas numéricas” en la página 52

Puede hacer referencia a una función como argumento de otra. Una función anidada se evalúa independientemente de la función externa (excepto cuando el compilador determina si una función mixta debe evaluarse utilizando instrucciones de coma fijo o coma flotante).

También puede anidar una expresión aritmética como argumento en una función numérica. Por ejemplo, en la sentencia siguiente, hay tres argumentos de función (a, by la expresión aritmética `(c / d)`):

```
Compute x = Function Sum(a b (c / d))
```

Puede hacer referencia a todos los elementos de una tabla (o matriz) como argumentos de función utilizando el subíndice `ALL` .

También puede utilizar los registros especiales enteros como argumentos siempre que se permitan argumentos enteros.

Conceptos relacionados
"Punto fijo contrastado con aritmética de coma flotante" en la página 54
Apéndice B, "Resultados intermedios y precisión aritmética", en la página 551

Referencias relacionadas
"HARIT" en la página 270

Ejemplos: funciones intrínsecas numéricas

Los siguientes ejemplos y las explicaciones que los acompañan muestran funciones intrínsecas en cada una de varias categorías.

Cuando los ejemplos siguientes muestran elementos de datos decimales con zona, en su lugar se podrían utilizar elementos decimales nacionales. (Sin embargo, los elementos decimales nacionales firmados requieren que la cláusula SIGN SEPARATE esté en vigor.)

Manejo de números general

Supongamos que desea encontrar el valor máximo de dos precios (representados a continuación como elementos alfanuméricos con signos de dólar), colocar este valor en un campo numérico en un registro de salida y determinar la longitud del registro de salida. Puede utilizar NUMVAL-C (una función que devuelve el valor numérico de un alfanumérico o literal nacional, o un elemento de datos alfanumérico o nacional) y las funciones MAX y LENGTH para hacerlo:

```
01  X                   Pic 9(2).
01  Price1              Pic x(8)   Value "$8000".
01  Price2              Pic x(8)   Value "$2000".
01  Output-Record.
    05  Product-Name    Pic x(20).
    05  Product-Number  Pic 9(9).
    05  Product-Price   Pic 9(6).
. . .
Procedure Division.
    Compute Product-Price =
      Function Max (Function Numval-C(Price1) Function Numval-C(Price2))
    Compute X = Function Length(Output-Record)
```

Además, para asegurarse de que el contenido de Product-Name está en mayúsculas, puede utilizar la sentencia siguiente:

```
Move Function Upper-case (Product-Name) to Product-Name
```

Fecha y hora

El ejemplo siguiente muestra cómo calcular una fecha de vencimiento que es 90 días a partir de hoy. Los primeros ocho caracteres devueltos por la función CURRENT-DATE representan la fecha en un formato de año de cuatro dígitos, mes de dos dígitos y día de dos dígitos (YYYYMMDD). La fecha se convierte a su valor entero; a continuación, se añade 90 a este valor y el entero se convierte de nuevo al formato YYYYMMDD .

```
01  YYYYMMDD         Pic 9(8).
01  Integer-Form     Pic S9(9).
. . .
    Move Function Current-Date(1:8) to YYYYMMDD
    Compute Integer-Form = Function Integer-of-Date(YYYYMMDD)
    Add 90 to Integer-Form
    Compute YYYYMMDD = Function Date-of-Integer(Integer-Form)
    Display 'Due Date: ' YYYYMMDD
```

Finanzas

Las decisiones de inversión empresarial con frecuencia requieren calcular el valor actual de los flujos de efectivo futuros esperados para evaluar la rentabilidad de una inversión planificada. El valor actual de una

cantidad que usted espera recibir en un momento dado en el futuro es esa cantidad, que, si se invierte hoy a un tipo de interés determinado, se acumularía a esa cantidad futura.

Por ejemplo, supongamos que una inversión propuesta de $1.000 produce una corriente de pago de $100, $200 y $300 en los próximos tres años, un pago por año respectivamente. Las siguientes sentencias COBOL calculan el valor actual de esas entradas de efectivo a un tipo de interés del 10%:

```
01  Series-Amt1      Pic 9(9)V99       Value 100.
01  Series-Amt2      Pic 9(9)V99       Value 200.
01  Series-Amt3      Pic 9(9)V99       Value 300.
01  Discount-Rate    Pic S9(2)V9(6)    Value .10.
01  Todays-Value     Pic 9(9)V99.
. . .
    Compute Todays-Value =
      Function
        Present-Value(Discount-Rate Series-Amt1 Series-Amt2 Series-Amt3)
```

Puede utilizar la función ANNUITY en problemas de negocio que requieren que determine el importe de un pago a plazos (anualidad) necesario para reembolsar el principal y el interés de un préstamo. La serie de pagos se caracteriza por una cantidad igual cada período, períodos de igual duración, y una tasa de interés igual cada período. El ejemplo siguiente muestra cómo puede calcular el pago mensual requerido para pagar un préstamo de $15.000 en tres años a una tasa de interés anual del 12% (36 pagos mensuales, intereses por mes = .12/12):

```
01  Loan             Pic 9(9)V99.
01  Payment          Pic 9(9)V99.
01  Interest         Pic 9(9)V99.
01  Number-Periods   Pic 99.
. . .
    Compute Loan = 15000
    Compute Interest = .12
    Compute Number-Periods = 36
    Compute Payment =
      Loan * Function Annuity((Interest / 12) Number-Periods)
```

Matemáticas

La siguiente sentencia COBOL demuestra que puede anidar funciones intrínsecas, utilizar expresiones aritméticas como argumentos y realizar cálculos complejos anteriormente simplemente:

```
Compute Z = Function Log(Function Sqrt (2 * X + 1)) + Function Rem(X 2)
```

Aquí en el addend la función intrínseca REM (en lugar de una sentencia DIVIDE con una cláusula REMAINDER) devuelve el resto de dividir X por 2.

Estadísticas

Las funciones intrínsecas facilitan el cálculo de la información estadística. Supongamos que está analizando varios impuestos de ciudad y desea calcular la media, la mediana y el rango (la diferencia entre los impuestos máximo y mínimo):

```
01  Tax-S            Pic 99v999 value .045.
01  Tax-T            Pic 99v999 value .02.
01  Tax-W            Pic 99v999 value .035.
01  Tax-B            Pic 99v999 value .03.
01  Ave-Tax          Pic 99v999.
01  Median-Tax       Pic 99v999.
01  Tax-Range        Pic 99v999.
. . .
    Compute Ave-Tax    = Function Mean   (Tax-S Tax-T Tax-W Tax-B)
    Compute Median-Tax = Function Median (Tax-S Tax-T Tax-W Tax-B)
    Compute Tax-Range  = Function Range  (Tax-S Tax-T Tax-W Tax-B)
```

Tareas relacionadas

“Conversión a números (NUMVAL, NUMVAL-C)” en la página 107

Punto fijo contrastado con aritmética de coma flotante

Cómo se codifica la aritmética en un programa (si una sentencia aritmética, una función intrínseca, una expresión o alguna combinación de estas anidadas entre sí) determina si la evaluación se realiza con coma flotante o aritmética de punto fijo.

Muchas sentencias en un programa podrían involucrar aritmética. Por ejemplo, cada uno de los siguientes tipos de sentencias COBOL requiere una evaluación aritmética:

- Aritmética general

```
compute report-matrix-col = (emp-count ** .5) + 1
add report-matrix-min to report-matrix-max giving report-matrix-tot
```

- Expresiones y funciones

```
compute report-matrix-col = function sqrt(emp-count) + 1
compute whole-hours       = function integer-part((average-hours) + 1)
```

- Comparaciones aritméticas

```
if report-matrix-col <     function sqrt(emp-count) + 1
if whole-hours       not = function integer-part((average-hours) + 1)
```

Evaluaciones de coma flotante

En general, si su codificación aritmética tiene cualquiera de las características enumeradas a continuación, se evalúa en aritmética de coma flotante:

- Un operando o campo de resultado es de coma flotante.

  Un operando es de coma flotante si lo codifica como un literal de coma flotante o si lo codifica como un elemento de datos definido como `USAGE COMP-1`, `USAGE COMP-2`, o coma flotante externa (`USAGE DISPLAY` o `USAGE NATIONAL` con un `PICTURE`de coma flotante).

  Un operando que es una expresión aritmética anidada o una referencia a una función intrínseca numérica da como resultado una aritmética de coma flotante cuando se cumple alguna de las condiciones siguientes:

  - Un argumento en una expresión aritmética da como resultado una coma flotante.
  - La función es una función de coma flotante.
  - La función es una función mixta con uno o más argumentos de coma flotante.
- Un exponente contiene posiciones decimales.

  Un exponente contiene posiciones decimales si utiliza un literal que contiene posiciones decimales, proporciona al elemento un `PICTURE` que contiene posiciones decimales o utiliza una expresión o función aritmética cuyo resultado tiene posiciones decimales.

Una expresión aritmética o una función numérica produce un resultado que tiene posiciones decimales si algún operando o argumento (excluyendo divisores y exponentes) tiene posiciones decimales.

Evaluaciones de punto fijo

En general, si una operación aritmética no contiene ninguna de las características listadas anteriormente para la coma flotante, el compilador hace que se evalúe en la aritmética de punto fijo. En otras palabras, las evaluaciones aritméticas se manejan como punto fijo sólo si todos los operandos son punto fijo, el campo de resultado se define como punto fijo y ninguno de los exponentes representa valores con posiciones decimales. Las expresiones aritméticas anidadas y las referencias de función también deben representar valores de punto fijo.

Comparaciones aritméticas (condiciones de relación)

Cuando compara expresiones numéricas utilizando un operador relacional, las expresiones numéricas (ya sean elementos de datos, expresiones aritméticas, referencias de función o alguna combinación de ellas) son comparados en el contexto de toda la evaluación. Es decir, los atributos de cada uno pueden influir en la evaluación del otro: ambas expresiones se evalúan en punto fijo, o ambas se evalúan en coma flotante. Esto también se aplica a las comparaciones abreviadas aunque una comparación no aparezca explícitamente en la comparación. Por ejemplo:

```
if (a + d) = (b + e) and c
```

Esta sentencia tiene dos comparaciones: `(a + d) = (b + e)` y `(a + d) = c`. Aunque `(a + d)` no aparece explícitamente en la segunda comparación, es una comparación en esa comparación. Por lo tanto, los atributos de c pueden influir en la evaluación de `(a + d)`.

El compilador maneja comparaciones (y la evaluación de cualquier expresión aritmética anidada en comparaciones) en aritmética de coma flotante si la comparación es un valor de coma flotante o se resuelve en un valor de coma flotante.

El compilador maneja comparaciones (y la evaluación de cualquier expresión aritmética anidada en comparaciones) en aritmética de punto fijo si ambas comparaciones son valores de punto fijo o se resuelven en valores de punto fijo.

Sin embargo, las comparaciones implícitas (no se utiliza ningún operador relacional) no se manejan como una unidad; las dos comparaciones se tratan por separado en cuanto a su evaluación en coma flotante o aritmética de punto fijo. En el ejemplo siguiente, cinco expresiones aritméticas se evalúan independientemente de los atributos del otro y, a continuación, se comparan entre sí.

```
evaluate (a + d)
    when (b + e) thru c
    when (f / g) thru (h * i)
    . . .
end-evaluate
```

“Ejemplos: evaluaciones de punto fijo y de coma flotante” en la página 55

Referencias relacionadas
“Expresiones aritméticas en sentencias no aritméticas” en la página 559

Ejemplos: evaluaciones de punto fijo y de coma flotante

El ejemplo siguiente muestra sentencias que se evalúan utilizando aritmética de punto fijo y utilizando aritmética de coma flotante.

Supongamos que define los elementos de datos para una tabla de empleados de la siguiente manera:

```
01  employee-table.
    05  emp-count         pic  9(4).
    05  employee-record occurs 1 to 1000 times
            depending on emp-count.
        10 hours           pic +9(5)ve+99.
. . .
01  report-matrix-col     pic  9(3).
01  report-matrix-min     pic  9(3).
01  report-matrix-max     pic  9(3).
01  report-matrix-tot     pic  9(3).
01  average-hours         pic  9(3)v9.
01  whole-hours           pic  9(4).
```

Estas sentencias se evalúan utilizando aritmética de coma flotante:

```
compute report-matrix-col = (emp-count ** .5) + 1
```

```
compute report-matrix-col = function sqrt(emp-count) + 1
if report-matrix-tot < function sqrt(emp-count) + 1
```

Estas sentencias se evalúan utilizando aritmética de punto fijo:

```
add report-matrix-min to report-matrix-max giving report-matrix-tot
compute report-matrix-max =
    function max(report-matrix-max report-matrix-tot)
if whole-hours not = function integer-part((average-hours) + 1)
```

Utilización de signos de moneda

Muchos programas necesitan procesar información financiera y presentar esa información utilizando los signos de moneda apropiados. Con el soporte de moneda COBOL (y la página de códigos adecuada para la impresora o unidad de pantalla), puede utilizar varios signos de moneda en un programa.

Acerca de esta tarea

Puede utilizar uno o varios de los signos siguientes:

- Símbolos como el signo de dólar ($)
- Signos de moneda de más de un carácter (como USD o EUR)
- Firma del euro, establecida por la Unión Económica y Monetaria (UEM)

Para especificar los símbolos para visualizar información financiera, utilice la cláusula CURRENCY SIGN (en el párrafo SPECIAL-NAMES de la CONFIGURATION SECTION) con los caracteres PICTURE relacionados con dichos símbolos. En el ejemplo siguiente, el PICTURE carácter $ indica que se va a utilizar el signo de moneda $US :

```
    Currency Sign is "$US" with Picture Symbol "$".
    . . .
77  Invoice-Amount      Pic $$,$$9.99.
. . .
    Display "Invoice amount is " Invoice-Amount.
```

En este ejemplo, si Invoice-Amount contenía 1500.00, la salida de visualización sería:

```
Invoice amount is  $US1,500.00
```

Al utilizar más de una cláusula CURRENCY SIGN en el programa, puede permitir que se muestren varios signos de moneda.

Puede utilizar un literal hexadecimal para indicar el valor de signo de moneda. El uso de un literal hexadecimal podría ser útil si el método de entrada de datos para el programa fuente no permite la entrada de los caracteres deseados fácilmente. El ejemplo siguiente muestra el valor hexadecimal X'80' utilizado como signo de moneda:

```
    Currency Sign X'80' with Picture Symbol 'U'.
    . . .
01  Deposit-Amount       Pic UUUUU9.99.
```

Si no hay ningún carácter correspondiente para el signo de euro en el teclado, debe especificarlo como un valor hexadecimal en la cláusula CURRENCY SIGN .

El valor hexadecimal para el signo de euro es X'80' con la página de códigos 1252 (Latin 1).

Referencias relacionadas
“MONEDA” en la página 277
Cláusula CURRENCY SIGN (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: varios signos de moneda

El ejemplo siguiente muestra cómo puede visualizar valores en euros (como EUR) y francos suizos (como CHF).

```
IDENTIFICATION DIVISION.
PROGRAM-ID. EuroSamp.
Environment Division.
Configuration Section.
Special-Names.
    Currency Sign is "CHF "  with Picture Symbol "F"
    Currency Sign is "EUR "  with Picture Symbol "U".
Data Division.
WORKING-STORAGE SECTION.
01  Deposit-in-Euro        Pic S9999V99 Value 8000.00.
01  Deposit-in-CHF         Pic S99999V99.
01  Deposit-Report.
    02  Report-in-Franc    Pic -FFFFF9.99.
    02  Report-in-Euro     Pic -UUUUU9.99.
01  EUR-to-CHF-Conv-Rate   Pic 9V99999  Value 1.53893.
. . .
PROCEDURE DIVISION.
Report-Deposit-in-CHF-and-EUR.
    Move Deposit-in-Euro to Report-in-Euro
    Compute Deposit-in-CHF Rounded
          = Deposit-in-Euro * EUR-to-CHF-Conv-Rate
      On Size Error
        Perform Conversion-Error
      Not On Size Error
        Move Deposit-in-CHF to Report-in-Franc
        Display "Deposit in euro  = " Report-in-Euro
        Display "Deposit in franc = " Report-in-Franc
    End-Compute
    Goback.
Conversion-Error.
           Display "Conversion error from EUR to CHF"
           Display "Euro value: " Report-in-Euro.
```

El ejemplo anterior genera la siguiente salida de visualización:

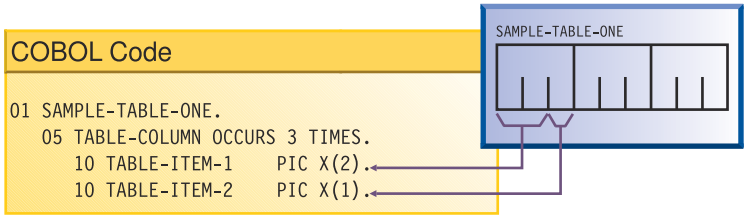
```
Deposit in euro  =  EUR 8000.00
Deposit in franc = CHF 12311.44
```

El tipo de cambio utilizado en este ejemplo es sólo para fines ilustrativos.

Capítulo 4. Manejo de tablas

Acerca de esta tarea

Una *tabla* es una colección de elementos de datos que tienen la misma descripción, como totales de cuenta o promedios mensuales. Una tabla consta de un nombre de tabla y elementos subordinados denominados *elementos de tabla*. Una tabla es el equivalente COBOL de una matriz.



En el ejemplo anterior, `SAMPLE-TABLE-ONE` es el elemento de grupo que contiene la tabla. `TABLE-COLUMN` nombra el elemento de tabla de una tabla unidimensional que aparece tres veces.

En lugar de definir elementos repetitivos como entradas consecutivas separadas en `DATA DIVISION`, utilice la cláusula `OCCURS` en la entrada `DATA DIVISION` para definir una tabla. Esta práctica tiene estas ventajas:

- El código muestra claramente la unidad de los elementos (los elementos de tabla).
- Puede utilizar subíndices e índices para hacer referencia a los elementos de tabla.
- Puede repetir fácilmente los elementos de datos.

Las tablas son importantes para aumentar la velocidad de un programa, especialmente un programa que busca registros.

Conceptos relacionados
"SE PRODUCE COMPLEJO EN FUNCIÓN DE" en la página 73

Tareas relacionadas
"Definición de una tabla (OCCURS)" en la página 59
"Anidamiento de tablas" en la página 60
"Cómo hacer referencia a un elemento de una tabla" en la página 62
"Colocación de valores en una tabla" en la página 65
"Creación de tablas de longitud variable (DEPENDING ON)" en la página 70
"Búsqueda de una tabla" en la página 77
"Proceso de elementos de tabla utilizando funciones intrínsecas" en la página 80

"Manejo eficiente de tablas" en la página 529

Definición de una tabla (OCCURS)

Para codificar una tabla, asigne a la tabla un nombre de grupo y defina un elemento subordinado (el elemento de tabla) que se repetirá *n* veces.

Acerca de esta tarea

```
01  table-name.
    05 element-name OCCURS n TIMES.
    . . . (subordinate items of the table element)
```

En el ejemplo anterior, `table-name` es el nombre de un elemento de grupo alfanumérico. La definición de elemento de tabla (que incluye la cláusula OCCURS) está subordinada al elemento de grupo que contiene la tabla. La cláusula OCCURS no se puede utilizar en una descripción level-01 .

Si una tabla debe contener sólo datos Unicode (UTF-16) y desea que el elemento de grupo que contiene la tabla se comporte como un elemento nacional de categoría elemental en la mayoría de las operaciones, codifique la cláusula `GROUP-USAGE NATIONAL` para el elemento de grupo:

```
01  table-nameN Group-Usage National.
    05  element-nameN OCCURS m TIMES.
        10  elementN1  Pic nn.
        10  elementN2  Pic S99 Sign Is Leading, Separate.
        . . .
```

Cualquier elemento elemental que esté subordinado a un grupo nacional debe describirse explícita o implícitamente como `USAGE NATIONAL`, y cualquier elemento de datos numérico subordinado que esté firmado debe estar implícita o explícitamente descrito con la cláusula `SIGN IS SEPARATE` .

Para crear tablas de dos a siete dimensiones, utilice cláusulas OCCURS anidadas.

Para crear una tabla de longitud variable, codifique la frase `DEPENDING ON` de la cláusula OCCURS .

Para especificar que los elementos de tabla se ordenarán en orden ascendente o descendente basándose en los valores de uno o más campos clave de la tabla, codifique las frases `ASCENDING` o `DESCENDING KEY` de la cláusula OCCURS , o ambas. Especifique los nombres de las claves en orden decreciente de significación. Las claves pueden ser de clase alfabética, alfanumérica, DBCS, nacional o numérica. (Si tiene `USAGE NATIONAL`, una clave puede ser de categoría nacional, o puede ser un elemento de coma flotante nacional, editado numérica-editado, decimal nacional o nacional.)

Debe codificar la frase `ASCENDING` o `DESCENDING KEY` de la cláusula OCCURS para realizar una búsqueda binaria (`SEARCH ALL`) de una tabla. Puede utilizar una sentencia SORT de formato 2 para ordenar la tabla de acuerdo con sus claves definidas, haciendo así que la tabla se pueda buscar mediante la sentencia `SEARCH ALL` . Tenga en cuenta que `SEARCH ALL` devolverá resultados imprevisibles si la tabla no se ha ordenado según las claves.

“Ejemplo: búsqueda binaria” en la página 79

Conceptos relacionados
“Grupos nacionales” en la página 198

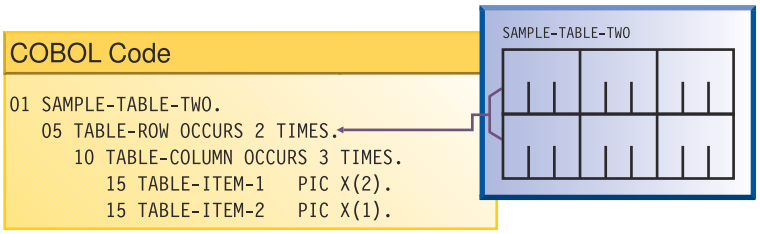
Tareas relacionadas
“Anidamiento de tablas” en la página 60
“Cómo hacer referencia a un elemento de una tabla” en la página 62
“Colocación de valores en una tabla” en la página 65
“Creación de tablas de longitud variable (DEPENDING ON)” en la página 70
“Utilización de grupos nacionales” en la página 202
“Realización de una búsqueda binaria (SEARCH ALL)” en la página 79
“Definición de datos numéricos” en la página 35

Referencias relacionadas
Cláusula OCCURS (*COBOL for Linux en x86 Consulta de lenguaje*)
Cláusula SIGN (*COBOL for Linux en x86 Consulta de lenguaje*)
Frases ASCENDENTE KEY y DESCENDENTE KEY
(*COBOL for Linux en x86 Consulta de lenguaje*)
Sentencia SORT (*COBOL for Linux en x86 Consulta de lenguaje*)

Anidamiento de tablas

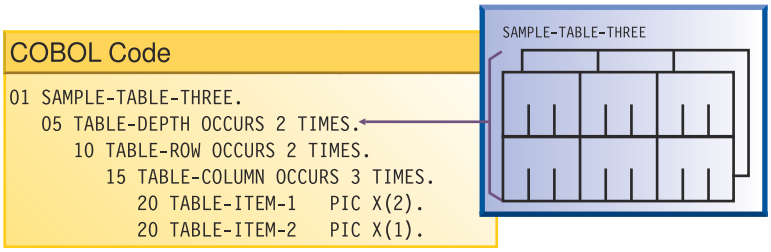
Para crear una tabla bidimensional, defina una tabla unidimensional en cada aparición de otra tabla unidimensional.

Acerca de esta tarea



Por ejemplo, en SAMPLE-TABLE-TWO anterior, TABLE-ROW es un elemento de una tabla unidimensional que se produce dos veces. TABLE-COLUMN es un elemento de una tabla bidimensional que aparece tres veces en cada aparición de TABLE-ROW.

Para crear una tabla tridimensional, defina una tabla unidimensional en cada aparición de otra tabla unidimensional, que a su vez está contenida en cada aparición de otra tabla unidimensional. Por ejemplo:



En SAMPLE-TABLE-THREE, TABLE-DEPTH es un elemento de una tabla unidimensional que se produce dos veces. TABLE-ROW es un elemento de una tabla bidimensional que se produce dos veces dentro de cada aparición de TABLE-DEPTH. TABLE-COLUMN es un elemento de una tabla tridimensional que se produce tres veces dentro de cada aparición de TABLE-ROW.

En una tabla bidimensional, los dos subíndices corresponden a los números de fila y columna. En una tabla tridimensional, los tres subíndices corresponden a los números de profundidad, fila y columna.

“Ejemplo: subscripción” en la página 61
“Ejemplo: indexación” en la página 62

Tareas relacionadas

“Definición de una tabla (OCCURS)” en la página 59
“Cómo hacer referencia a un elemento de una tabla” en la página 62
“Colocación de valores en una tabla” en la página 65
“Creación de tablas de longitud variable (DEPENDING ON)” en la página 70
“Búsqueda de una tabla” en la página 77
“Proceso de elementos de tabla utilizando funciones intrínsecas” en la página 80
“Manejo eficiente de tablas” en la página 529

Referencias relacionadas

Cláusula OCCURS (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: subscripción

El ejemplo siguiente muestra referencias válidas a SAMPLE-TABLE-THREE que utilizan subíndices literales. Los espacios son necesarios en el segundo ejemplo.

```
TABLE-COLUMN (2, 2, 1)
TABLE-COLUMN (2 2 1)
```

En cualquier referencia de tabla, el primer valor (2) hace referencia a la segunda aparición dentro de TABLE-DEPTH, el segundo valor (2) hace referencia a la segunda aparición dentro de TABLE-ROWy el tercer valor (1) hace referencia a la primera aparición dentro de TABLE-COLUMN.

La referencia siguiente a SAMPLE-TABLE-TWO utiliza subíndices de variables. La referencia es válida si SUB1 y SUB2 son nombres de datos que contienen valores enteros positivos dentro del rango de la tabla.

```
TABLE-COLUMN (SUB1 SUB2)
```

Tareas relacionadas
"Suscripción" en la página 63

Ejemplo: indexación

El ejemplo siguiente muestra cómo se calculan los desplazamientos a elementos a los que se hace referencia con índices.

Considere la siguiente tabla tridimensional, SAMPLE-TABLE-FOUR:

```
01  SAMPLE-TABLE-FOUR
    05 TABLE-DEPTH OCCURS 3 TIMES INDEXED BY INX-A.
       10 TABLE-ROW OCCURS 4 TIMES INDEXED BY INX-B.
          15 TABLE-COLUMN OCCURS 8 TIMES INDEXED BY INX-C  PIC X(8).
```

Supongamos que codifica la siguiente referencia de indexación relativa a SAMPLE-TABLE-FOUR:

```
TABLE-COLUMN (INX-A + 1, INX-B + 2, INX-C - 1)
```

Esta referencia provoca el siguiente cálculo del desplazamiento al elemento TABLE-COLUMN :

```
  (contents of INX-A) + (256 * 1)
+ (contents of INX-B) + (64 * 2)
+ (contents of INX-C) - (8 * 1)
```

Este cálculo se basa en las siguientes longitudes de elemento:

- Cada aparición de TABLE-DEPTH tiene 256 bytes de longitud (4 * 8 * 8).
- Cada aparición de TABLE-ROW tiene 64 bytes de longitud (8 * 8).
- Cada aparición de TABLE-COLUMN tiene una longitud de 8 bytes.

Tareas relacionadas
"Indexación" en la página 63

Cómo hacer referencia a un elemento de una tabla

Un elemento de tabla tiene un nombre de colectivo, pero los elementos individuales que contiene no tienen nombres de datos exclusivos.

Acerca de esta tarea

Para hacer referencia a un elemento, puede elegir entre tres técnicas:

- Utilice el nombre de datos del elemento de tabla, junto con su número de aparición (denominado *subíndice*) entre paréntesis. Esta técnica se denomina *suscripción*.
- Utilice el nombre de datos del elemento de tabla, junto con un valor (denominado *índice*) que se añade a la dirección de la tabla para localizar un elemento (como desplazamiento desde el principio de la tabla). Esta técnica se denomina *indexación*o suscripción utilizando nombres de índice.

- Utilice subíndices e índices juntos.

Tareas relacionadas
“Suscripción” en la página 63
“Indexación” en la página 63

Suscripción

El valor de subíndice más bajo posible es 1, que hace referencia a la primera aparición de un elemento de tabla. En una tabla unidimensional, el subíndice corresponde al número de fila.

Acerca de esta tarea

Puede utilizar un literal o un nombre de datos como subíndice. Si un elemento de datos que tiene un subíndice literal es de longitud fija, el compilador resuelve la ubicación del elemento de datos.

Cuando utilice un nombre de datos como subíndice de variable, debe describir el nombre de datos como un entero numérico elemental. El formato más eficiente es `COMPUTATIONAL` (`COMP`) con un tamaño de `PICTURE` que es menor que cinco dígitos. No puede utilizar un subíndice con un nombre de datos que se utilice como subíndice. El código generado para la aplicación resuelve la ubicación de un subíndice de variable en tiempo de ejecución.

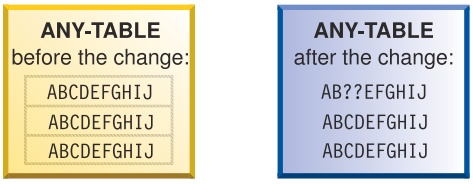
Puede incrementar o disminuir un subíndice literal o variable en una cantidad entera especificada. Por ejemplo:

```
TABLE-COLUMN (SUB1 - 1, SUB2 + 3)
```

Puede cambiar parte de un elemento de tabla en lugar de todo el elemento. Para ello, consulte la posición de carácter y la longitud de la subserie que se va a cambiar. Por ejemplo:

```
01  ANY-TABLE.
    05 TABLE-ELEMENT     PIC X(10)
        OCCURS 3 TIMES   VALUE "ABCDEFGHIJ".
. . .
    MOVE "??" TO TABLE-ELEMENT (1) (3 : 2).
```

La sentencia `MOVE` del ejemplo anterior mueve la serie '??' al número de elemento de tabla 1, empezando en la posición de carácter 3, para una longitud de 2 caracteres.



“Ejemplo: subscripción” en la página 61

Tareas relacionadas
“Indexación” en la página 63
“Colocación de valores en una tabla” en la página 65
“Búsqueda de una tabla” en la página 77
“Manejo eficiente de tablas” en la página 529

Indexación

Puede crear un índice utilizando la frase `INDEXED BY` de la cláusula `OCCURS` para identificar un nombre de índice.

Acerca de esta tarea

Por ejemplo, INX-A en el código siguiente es un nombre de índice:

```
05 TABLE-ITEM PIC X(8)
     OCCURS 10 INDEXED BY INX-A.
```

El compilador calcula el valor contenido en el índice como el número de aparición (subíndice) menos 1, multiplicado por la longitud del elemento de tabla. Por lo tanto, para la quinta aparición de TABLE-ITEM, el valor binario contenido en INX-A es (5-1) * 8 o 32.

Puede utilizar un nombre de índice para hacer referencia a otra tabla sólo si ambas descripciones de tabla tienen el mismo número de elementos de tabla y los elementos de tabla tienen la misma longitud.

Puede utilizar la cláusula USAGE IS INDEX para crear un elemento de datos de índice y puede utilizar un elemento de datos de índice con cualquier tabla. Por ejemplo, INX-B en el código siguiente es un elemento de datos de índice:

```
77  INX-B  USAGE IS INDEX.
. . .
    SET INX-A TO 10
    SET INX-B TO INX-A.
    PERFORM VARYING INX-A FROM 1 BY 1 UNTIL INX-A > INX-B
        DISPLAY TABLE-ITEM (INX-A)
        . . .
    END-PERFORM.
```

El nombre de índice INX-A se utiliza para atravesar la tabla TABLE-ITEM anterior. El elemento de datos de índice INX-B se utiliza para contener el índice del último elemento de la tabla. La ventaja de este tipo de codificación es que el cálculo de desplazamientos de elementos de tabla se minimiza y no es necesaria ninguna conversión para la condición UNTIL .

Puede utilizar la sentencia SET para asignar a un elemento de datos de índice el valor que ha almacenado en un nombre-índice, como en la sentencia SET INX-B TO INX-A anterior. Por ejemplo, cuando carga registros en una tabla de longitud variable, puede almacenar el valor de índice del último registro en un elemento de datos definido como USAGE IS INDEX. A continuación, puede probar el final de la tabla comparando el valor de índice actual con el valor de índice del último registro. Esta técnica es útil cuando se mira a través o se procesa una tabla.

Puede incrementar o disminuir un nombre de índice mediante un elemento de datos entero elemental o un literal entero distinto de cero, por ejemplo:

```
SET INX-A DOWN BY 3
```

El entero representa un número de apariciones. Se convierte en un valor de índice antes de añadirse o restarse del índice.

Inicialice el nombre de índice utilizando una sentencia SET, PERFORM VARYINGo SEARCH ALL . A continuación, puede utilizar el nombre de índice en SEARCH o sentencias de condición relacional. Para cambiar el valor, utilice una sentencia PERFORM, SEARCHo SET .

Puesto que está comparando un desplazamiento físico, puede utilizar directamente elementos de datos de índice sólo en sentencias SEARCH y SET o en comparaciones con índices u otros elementos de datos de índice. No puede utilizar elementos de datos de índice como subíndices o índices.

“Ejemplo: indexación” en la página 62

Tareas relacionadas

“Suscripción” en la página 63
“Colocación de valores en una tabla” en la página 65
“Búsqueda de una tabla” en la página 77
“Proceso de elementos de tabla utilizando funciones intrínsecas” en la página 80
“Manejo eficiente de tablas” en la página 529

Referencias relacionadas
frase INDEXED BY (*COBOL for Linux en x86 Consulta de lenguaje*)
frase INDEX (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia SET (*COBOL for Linux en x86 Consulta de lenguaje*)

Colocación de valores en una tabla

Puede colocar valores en una tabla cargando la tabla dinámicamente, inicializando la tabla con la sentencia `INITIALIZE` o asignando valores con la cláusula `VALUE` al definir la tabla.

Acerca de esta tarea

Tareas relacionadas
“Carga dinámica de una tabla” en la página 65
“Carga de una tabla de longitud variable” en la página 72
“Inicialización de una tabla (INITIALIZE)” en la página 65
“Asignación de valores al definir una tabla (VALUE)” en la página 66
“Asignación de valores a una tabla de longitud variable” en la página 73

Carga dinámica de una tabla

Si los valores iniciales de una tabla son diferentes con cada ejecución del programa, puede definir la tabla sin valores iniciales. En su lugar, puede leer los valores cambiados en la tabla dinámicamente antes de que el programa haga referencia a la tabla.

Acerca de esta tarea

Para cargar una tabla, utilice la sentencia `PERFORM` y el subíndice o la indexación.

Al leer datos para cargar la tabla, realice una prueba para asegurarse de que los datos no exceden el espacio asignado para la tabla. Utilice un valor con nombre (en lugar de un literal) para el recuento máximo de elementos. A continuación, si hace que la tabla sea más grande, sólo tendrá que cambiar un valor en lugar de todas las referencias a un literal.

“Ejemplo: PERFORM y subscripción” en la página 69
“Ejemplo: PERFORM e indexación” en la página 69

Referencias relacionadas
sentencia PERFORM (*COBOL for Linux en x86 Consulta de lenguaje*)

Inicialización de una tabla (INITIALIZE)

Puede cargar una tabla codificando una o más sentencias `INITIALIZE` .

Acerca de esta tarea

Por ejemplo, para mover el valor 3 a cada uno de los elementos de datos numéricos elementales de una tabla denominada `TABLE-ONE`, que se muestra a continuación, puede codificar la sentencia siguiente:

```
INITIALIZE TABLE-ONE REPLACING NUMERIC DATA BY 3.
```

Para mover el carácter 'X' a cada uno de los elementos de datos alfanuméricos elementales en `TABLE-ONE`, puede codificar la sentencia siguiente:

```
INITIALIZE TABLE-ONE REPLACING ALPHANUMERIC DATA BY "X".
```

Cuando se utiliza la sentencia `INITIALIZE` para inicializar una tabla, la tabla se procesa como un elemento de grupo (es decir, con semántica de grupo); los elementos de datos elementales del grupo se

reconocen y se procesan. Por ejemplo, supongamos que TABLE-ONE es un grupo alfanumérico que se define de este modo:

```
01  TABLE-ONE.
  02  Trans-out  Occurs 20.
      05  Trans-code       Pic X    Value "R".
      05  Part-number      Pic XX   Value "13".
      05  Trans-quan       Pic 99   Value 10.
      05  Price-fields.
          10  Unit-price   Pic 99V  Value 50.
          10  Discount     Pic 99V  Value 25.
          10  Sales-Price  Pic 999  Value 375.
      . . .
      Initialize TABLE-ONE Replacing Numeric Data By 3
                                     Alphanumeric Data By "X"
```

La tabla siguiente muestra el contenido que cada uno de los veinte elementos de 12 bytes Trans-out(*n*) tiene antes de la ejecución y después de la ejecución de la sentencia INITIALIZE mostrada anteriormente:

Trans-out(*n*) antes	**Trans-out(*n*) después de**
R13105025375	XX*b*030303003[1]
1. El símbolo *b* representa un espacio en blanco.	

De forma similar, puede utilizar una sentencia INITIALIZE para cargar una tabla definida como un grupo nacional. Por ejemplo, si TABLE-ONE mostrada anteriormente especificaba la cláusula GROUP-USAGE NATIONAL , y Trans-code y Part-number tenían N en lugar de X en sus cláusulas PICTURE , la sentencia siguiente tendría el mismo efecto que la sentencia INITIALIZE anterior, excepto que los datos de TABLE-ONE se codificarían en su lugar en UTF-16:

```
Initialize TABLE-ONE Replacing Numeric  Data By 3
                               National Data By N"X"
```

La frase REPLACING NUMERIC también inicializa elementos de datos de coma flotante.

Puede utilizar la frase REPLACING de la sentencia INITIALIZE de forma similar para inicializar todos los elementos elementales ALPHABETIC, DBCS, ALPHANUMERIC-EDITED, NATIONAL-EDITED, y NUMERIC-EDITED elementos de datos de una tabla.

La sentencia INITIALIZE no puede asignar valores a una tabla de longitud variable (es decir, una tabla que se ha definido utilizando la cláusula OCCURS DEPENDING ON).

"Ejemplos: inicialización de elementos de datos" en la página 24

Tareas relacionadas

"Inicialización de una estructura (INITIALIZE)" en la página 27
"Asignación de valores al definir una tabla (VALUE)" en la página 66
"Asignación de valores a una tabla de longitud variable" en la página 73
"Bucle a través de una tabla" en la página 93
"Utilización de elementos de datos y elementos de grupo" en la página 20
"Utilización de grupos nacionales" en la página 202

Referencias relacionadas

sentencia INITIALIZE (*COBOL for Linux en x86 Consulta de lenguaje*)

Asignación de valores al definir una tabla (VALUE)

Si una tabla va a contener valores estables (como días y meses), puede establecer los valores específicos cuando defina la tabla.

Acerca de esta tarea

Establezca los valores estáticos en las tablas de una de estas maneras:

- Inicialice cada elemento de tabla individualmente.
- Inicialice una tabla completa a nivel de grupo .
- Inicialice todas las apariciones de un elemento de tabla determinado con el mismo valor.

Tareas relacionadas
“Inicializando cada elemento de tabla individualmente” en la página 67
“Inicialización de una tabla a nivel de grupo” en la página 68
“Inicialización de todas las apariciones de un elemento de tabla determinado” en la página 68
“Inicialización de una estructura (INITIALIZE)” en la página 27

Inicializando cada elemento de tabla individualmente

Si una tabla es pequeña, puede establecer el valor de cada elemento individualmente utilizando una cláusula VALUE .

Acerca de esta tarea

Utilice la técnica siguiente, que se muestra en el código de ejemplo siguiente:

Procedimiento

1. Defina un registro (como, por ejemplo, `Error-Flag-Table` a continuación) que contiene los elementos que van a estar en la tabla.
2. Establezca el valor inicial de cada elemento en una cláusula VALUE .
3. Codifique una entrada REDEFINES para convertir el registro en una tabla.

Resultados

```
*************************************************************
***           E R R O R   F L A G   T A B L E             ***
*************************************************************
 01  Error-Flag-Table                   Value Spaces.
   88 No-Errors                         Value Spaces.
     05 Type-Error                      Pic X.
     05 Shift-Error                     Pic X.
     05 Home-Code-Error                 Pic X.
     05 Work-Code-Error                 Pic X.
     05 Name-Error                      Pic X.
     05 Initials-Error                  Pic X.
     05 Duplicate-Error                 Pic X.
     05 Not-Found-Error                 Pic X.
 01  Filler Redefines Error-Flag-Table.
     05 Error-Flag Occurs 8 Times
          Indexed By Flag-Index         Pic X.
```

En el ejemplo anterior, la cláusula VALUE en el nivel 01 inicializa cada uno de los elementos de tabla con el mismo valor. En su lugar, cada elemento de tabla podría describirse con su propia cláusula VALUE para inicializar ese elemento en un valor distinto.

Para inicializar tablas más grandes, utilice sentencias MOVE, PERFORMo INITIALIZE .

Tareas relacionadas
“Inicialización de una estructura (INITIALIZE)” en la página 27
“Asignación de valores a una tabla de longitud variable” en la página 73

Referencias relacionadas
Cláusula REDEFINES (*COBOL for Linux en x86 Consulta de lenguaje*)
Cláusula OCCURS (*COBOL for Linux en x86 Consulta de lenguaje*)

Inicialización de una tabla a nivel de grupo

Codifique un elemento de datos de grupo nacional o alfanumérico y asígnele, a través de la cláusula VALUE , el contenido de toda la tabla. A continuación, en un elemento de datos subordinado, utilice una cláusula OCCURS para definir los elementos de tabla individuales.

Acerca de esta tarea

En el ejemplo siguiente, el elemento de datos de grupo alfanumérico TABLE-ONE utiliza una cláusula VALUE que inicializa cada uno de los cuatro elementos de TABLE-TWO:

```
01  TABLE-ONE                     VALUE "1234".
    05 TABLE-TWO OCCURS 4 TIMES   PIC X.
```

En el ejemplo siguiente, el elemento de datos de grupo nacional Table-OneN utiliza una cláusula VALUE que inicializa cada uno de los tres elementos del elemento de datos subordinado Table-TwoN (cada uno de los cuales es implícitamente USAGE NATIONAL). Tenga en cuenta que puede inicializar un elemento de datos de grupo nacional con una cláusula VALUE que utilice un literal alfanumérico, tal como se muestra a continuación, o un literal nacional.

```
01  Table-OneN  Group-Usage National  Value "AB12CD34EF56".
    05  Table-TwoN   Occurs 3 Times   Indexed By MyI.
        10  ElementOneN  Pic nn.
        10  ElementTwoN  Pic 99.
```

Después de inicializar Table-OneN , ElementOneN(1) contiene NX"41004200" (la representación UTF-16 de 'AB'), el elemento decimal nacional ElementTwoN(1) contiene NX"31003200" (la representación UTF-16 de'12'), etc.

Referencias relacionadas
Cláusula OCCURS (*COBOL for Linux en x86 Consulta de lenguaje*)
cláusula GROUP-USAGE (*COBOL for Linux en x86 Consulta de lenguaje*)

Inicialización de todas las apariciones de un elemento de tabla determinado

Puede utilizar la cláusula VALUE en la descripción de datos de un elemento de tabla para inicializar todas las instancias de ese elemento en el valor especificado.

Acerca de esta tarea

```
01  T2.
    05 T-OBJ                   PIC 9   VALUE 3.
    05 T OCCURS 5 TIMES
           DEPENDING ON T-OBJ.
       10 X                    PIC XX  VALUE "AA".
       10 Y                    PIC 99  VALUE 19.
       10 Z                    PIC XX  VALUE "BB".
```

Por ejemplo, el código anterior hace que todos los elementos X (1 a 5) se inicialicen en AA, que todos los elementos Y (1 a 5) se inicialicen en 19 y que todos los elementos Z (1 a 5) se inicialicen en BB. A continuación, T-OBJ se establece en 3.

Tareas relacionadas
“Asignación de valores a una tabla de longitud variable” en la página 73

Referencias relacionadas
Cláusula OCCURS (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: PERFORM y subscripción

Este ejemplo atraviesa una tabla de distintivos de error utilizando la suscripción hasta que se encuentra un código de error que se ha establecido. Si se encuentra un código de error, el mensaje de error correspondiente se mueve a un campo de informe de impresión.

```
************************************************************
***          E R R O R   F L A G   T A B L E             ***
************************************************************
 01  Error-Flag-Table                  Value Spaces.
   88 No-Errors                       Value Spaces.
     05 Type-Error                    Pic X.
     05 Shift-Error                   Pic X.
     05 Home-Code-Error               Pic X.
     05 Work-Code-Error               Pic X.
     05 Name-Error                    Pic X.
     05 Initials-Error                Pic X.
     05 Duplicate-Error               Pic X.
     05 Not-Found-Error               Pic X.
 01  Filler Redefines Error-Flag-Table.
     05 Error-Flag Occurs 8 Times
          Indexed By Flag-Index       Pic X.
 77  Error-on                         Pic X  Value "E".
************************************************************
***         E R R O R   M E S S A G E   T A B L E        ***
************************************************************
 01  Error-Message-Table.
     05  Filler                        Pic X(25) Value
          "Transaction Type Invalid".
     05  Filler                        Pic X(25) Value
          "Shift Code Invalid".
     05  Filler                        Pic X(25) Value
          "Home Location Code Inval.".
     05  Filler                        Pic X(25) Value
          "Work Location Code Inval.".
     05  Filler                        Pic X(25) Value
          "Last Name - Blanks".
     05  Filler                        Pic X(25) Value
          "Initials - Blanks".
     05  Filler                        Pic X(25) Value
          "Duplicate Record Found".
     05  Filler                        Pic X(25) Value
          "Commuter Record Not Found".
 01  Filler Redefines Error-Message-Table.
     05  Error-Message Occurs 8 Times
           Indexed By Message-Index     Pic X(25).
 . . .
 PROCEDURE DIVISION.
     . . .
     Perform
          Varying Sub From 1 By 1
          Until No-Errors
        If Error-Flag (Sub) = Error-On
          Move Space To Error-Flag (Sub)
          Move Error-Message (Sub) To Print-Message
          Perform 260-Print-Report
        End-If
     End-Perform
     . . .
```

Ejemplo: PERFORM e indexación

Este ejemplo recorre una tabla de distintivos de error utilizando la indexación hasta que se encuentra un código de error que se ha establecido. Si se encuentra un código de error, el mensaje de error correspondiente se mueve a un campo de informe de impresión.

```
************************************************************
***          E R R O R   F L A G   T A B L E             ***
************************************************************
 01  Error-Flag-Table                  Value Spaces.
   88 No-Errors                       Value Spaces.
     05 Type-Error                    Pic X.
     05 Shift-Error                   Pic X.
     05 Home-Code-Error               Pic X.
```

```
     05 Work-Code-Error                 Pic X.
     05 Name-Error                      Pic X.
     05 Initials-Error                  Pic X.
     05 Duplicate-Error                 Pic X.
     05 Not-Found-Error                 Pic X.
 01  Filler Redefines Error-Flag-Table.
     05 Error-Flag Occurs 8 Times
          Indexed By Flag-Index         Pic X.
 77  Error-on                          Pic X  Value "E".
************************************************************
***         E R R O R   M E S S A G E   T A B L E        ***
************************************************************
 01  Error-Message-Table.
     05  Filler                          Pic X(25) Value
          "Transaction Type Invalid".
     05  Filler                          Pic X(25) Value
          "Shift Code Invalid".
     05  Filler                          Pic X(25) Value
          "Home Location Code Inval.".
     05  Filler                          Pic X(25) Value
          "Work Location Code Inval.".
     05  Filler                          Pic X(25) Value
          "Last Name - Blanks".
     05  Filler                          Pic X(25) Value
          "Initials - Blanks".
     05  Filler                          Pic X(25) Value
          "Duplicate Record Found".
     05  Filler                          Pic X(25) Value
          "Commuter Record Not Found".
 01  Filler Redefines Error-Message-Table.
     05  Error-Message Occurs 8 Times
           Indexed By Message-Index      Pic X(25).
 . . .
 PROCEDURE DIVISION.
     . . .
     Set Flag-Index To 1
     Perform Until No-Errors
       Search Error-Flag
         When Error-Flag (Flag-Index) = Error-On
           Move Space To Error-Flag (Flag-Index)
           Set Message-Index To Flag-Index
           Move Error-Message (Message-Index) To
             Print-Message
           Perform 260-Print-Report
       End-Search
     End-Perform
     . . .
```

Creación de tablas de longitud variable (DEPENDING ON)

Si no sabe antes del tiempo de ejecución cuántas veces se produce un elemento de tabla, defina una tabla de longitud variable. Para ello, utilice la cláusula `OCCURS DEPENDING ON` (ODO).

Acerca de esta tarea

```
X OCCURS 1 TO 10 TIMES DEPENDING ON Y
```

En el ejemplo anterior, X se denomina *sujeto ODO*y Y se denomina *objeto ODO*.

Dos factores afectan a la manipulación satisfactoria de los registros de longitud variable:

- Cálculo correcto de longitudes de registro

  La longitud de las partes variables de un elemento de grupo es el producto del objeto de la frase `DEPENDING ON` y la longitud del sujeto de la cláusula `OCCURS` .
- Conformidad de los datos del objeto de la cláusula `OCCURS DEPENDING ON` con su cláusula `PICTURE`

  Si el contenido del objeto ODO no coincide con su cláusula `PICTURE` , el programa podría terminar de forma anómala. Debe asegurarse de que el objeto ODO especifica correctamente el número actual de apariciones de elementos de tabla.

El ejemplo siguiente muestra un elemento de grupo (REC-1) que contiene el asunto y el objeto de la cláusula OCCURS DEPENDING ON . La forma en que se determina la longitud del elemento de grupo depende de si está enviando o recibiendo datos.

```
WORKING-STORAGE SECTION.
01  MAIN-AREA.
    03 REC-1.
        05 FIELD-1                       PIC 9.
        05 FIELD-2 OCCURS 1 TO 5 TIMES
           DEPENDING ON FIELD-1         PIC X(05).
01  REC-2.
    03 REC-2-DATA                       PIC X(50).
```

Si desea mover REC-1 (el elemento de envío en este caso) a REC-2, la longitud de REC-1 se determina inmediatamente antes del movimiento, utilizando el valor actual en FIELD-1. Si el contenido de FIELD-1 se ajusta a su cláusula PICTURE (es decir, si FIELD-1 contiene un elemento decimal con zona), el movimiento puede continuar basándose en la longitud real de REC-1. De lo contrario, el resultado es imprevisible. Debe asegurarse de que el objeto ODO tenga el valor correcto antes de iniciar el movimiento.

Cuando se realiza un movimiento a REC-1 (el elemento receptor en este caso), la longitud de REC-1 se determina utilizando el número máximo de apariciones. En este ejemplo, cinco apariciones de FIELD-2, más FIELD-1, producen una longitud de 26 bytes. En este caso, no es necesario establecer el objeto ODO (FIELD-1) antes de hacer referencia a REC-1 como elemento receptor. Sin embargo, el objeto ODO del campo emisor (no se muestra) debe establecerse en un valor numérico válido entre 1 y 5 para que el objeto ODO del campo receptor se establezca de forma válida mediante el movimiento.

Sin embargo, si realiza un movimiento a REC-1 (de nuevo el elemento receptor) donde REC-1 va seguido de un grupo de ubicación variable (un tipo de *ODO complejo*), la longitud real de REC-1 se calcula inmediatamente antes del movimiento, utilizando el valor actual del objeto ODO (FIELD-1). En el ejemplo siguiente, REC-1 y REC-2 están en el mismo registro, pero REC-2 no está subordinado a REC-1 y, por lo tanto, se encuentra de forma variable:

```
01  MAIN-AREA
    03 REC-1.
       05 FIELD-1                       PIC 9.
       05 FIELD-3                       PIC 9.
       05 FIELD-2 OCCURS 1 TO 5 TIMES
            DEPENDING ON FIELD-1        PIC X(05).
    03 REC-2.
       05 FIELD-4 OCCURS 1 TO 5 TIMES
            DEPENDING ON FIELD-3        PIC X(05).
```

El compilador emite un mensaje que le permite saber que se ha utilizado la longitud real. Este caso requiere que establezca el valor del objeto ODO antes de utilizar el elemento de grupo como campo de recepción.

El ejemplo siguiente muestra cómo definir una tabla de longitud variable cuando el objeto ODO (LOCATION-TABLE-LENGTH a continuación) está fuera del grupo:

```
 DATA DIVISION.
 FILE SECTION.
 FD  LOCATION-FILE.
 01  LOCATION-RECORD.
     05  LOC-CODE                 PIC XX.
     05  LOC-DESCRIPTION          PIC X(20).
     05  FILLER                   PIC X(58).
 WORKING-STORAGE SECTION.
 01  FLAGS.
     05 LOCATION-EOF-FLAG         PIC X(5) VALUE SPACE.
        88 LOCATION-EOF              VALUE "FALSE".
 01  MISC-VALUES.
     05 LOCATION-TABLE-LENGTH     PIC 9(3) VALUE ZERO.
     05 LOCATION-TABLE-MAX        PIC 9(3) VALUE 100.
```

```
****************************************************************
***               L O C A T I O N   T A B L E                ***
***               FILE CONTAINS LOCATION CODES.              ***
****************************************************************
 01  LOCATION-TABLE.
     05 LOCATION-CODE OCCURS 1 TO 100 TIMES
          DEPENDING ON LOCATION-TABLE-LENGTH   PIC X(80).
```

Conceptos relacionados
“SE PRODUCE COMPLEJO EN FUNCIÓN DE” en la página 73

Tareas relacionadas
“Asignación de valores a una tabla de longitud variable” en la página 73
“Carga de una tabla de longitud variable” en la página 72
“Cómo evitar la superposición al añadir elementos a una tabla de variables” en la página 76
“Búsqueda de la longitud de los elementos de datos” en la página 112

Referencias relacionadas
OCCURS DEPENDING ON cláusula
(*COBOL for Linux en x86 Consulta de lenguaje*)
Tablas de longitud variable (*COBOL for Linux en x86 Consulta de lenguaje*)

Carga de una tabla de longitud variable

Puede utilizar una estructura *do-until* (un bucle TEST AFTER) para controlar la carga de una tabla de longitud variable. Por ejemplo, después de ejecutar el código siguiente, LOCATION-TABLE-LENGTH contiene el subíndice del último elemento de la tabla.

Acerca de esta tarea

```
 DATA DIVISION.
 FILE SECTION.
 FD  LOCATION-FILE.
 01  LOCATION-RECORD.
     05  LOC-CODE              PIC XX.
     05  LOC-DESCRIPTION       PIC X(20).
     05  FILLER                PIC X(58).
 . . .
 WORKING-STORAGE SECTION.
 01  FLAGS.
     05 LOCATION-EOF-FLAG      PIC X(5) VALUE SPACE.
        88 LOCATION-EOF                 VALUE "YES".
 01  MISC-VALUES.
     05 LOCATION-TABLE-LENGTH  PIC 9(3) VALUE ZERO.
     05 LOCATION-TABLE-MAX     PIC 9(3) VALUE 100.
****************************************************************
***               L O C A T I O N   T A B L E                ***
***               FILE CONTAINS LOCATION CODES.              ***
****************************************************************
 01  LOCATION-TABLE.
     05 LOCATION-CODE OCCURS 1 TO 100 TIMES
          DEPENDING ON LOCATION-TABLE-LENGTH   PIC X(80).
 . . .
 PROCEDURE DIVISION.
     . . .
     Perform Test After
         Varying Location-Table-Length From 1 By 1
           Until Location-EOF
           Or Location-Table-Length = Location-Table-Max
       Move Location-Record To
           Location-Code (Location-Table-Length)
       Read Location-File
           At End Set Location-EOF To True
       End-Read
     End-Perform
```

Asignación de valores a una tabla de longitud variable

Puede codificar una VALUE cláusula para un elemento de grupo nacional o alfanumérico que tenga un elemento de datos subordinado que contenga la cláusula OCCURS con la frase DEPENDING ON . Cada estructura subordinada que contiene la frase DEPENDING ON se inicializa utilizando el número máximo de apariciones.

Acerca de esta tarea

Si define toda la tabla utilizando la frase DEPENDING ON , todos los elementos se inicializan utilizando el valor máximo definido del objeto ODO (OCCURS DEPENDING ON).

Si el objeto ODO se inicializa mediante una cláusula VALUE , se inicializa lógicamente después de que se haya inicializado el sujeto ODO.

```
01  TABLE-THREE           VALUE "3ABCDE".
    05 X                  PIC 9.
    05 Y OCCURS 5 TIMES
          DEPENDING ON X  PIC X.
```

Por ejemplo, en el código anterior, el asunto ODO Y(1) se inicializa en 'A', Y(2) en 'B',. ., Y(5) a 'E', y finalmente el objeto ODO X se inicializa en 3. Cualquier referencia posterior a TABLE-THREE (como en una sentencia DISPLAY) hace referencia a X y los tres primeros elementos, Y(1) a Y(3), de la tabla.

Tareas relacionadas
“Asignación de valores al definir una tabla (VALUE)” en la página 66

Referencias relacionadas
OCCURS DEPENDING ON cláusula
(*COBOL for Linux en x86 Consulta de lenguaje*)

SE PRODUCE COMPLEJO EN FUNCIÓN DE

Son posibles varios tipos de OCCURS DEPENDING ON complejos (*ODO complejo*). El ODO complejo está soportado como una extensión del 85 COBOL Estándar.

Las formas básicas de ODO complejo permitidas por el compilador son las siguientes:

- Elemento o grupo de ubicación variable: un elemento de datos descrito por una cláusula OCCURS con la frase DEPENDING ON va seguido de un elemento de datos elemental o de grupo no subordinado.
- Tabla de ubicación variable: un elemento de datos descrito por una cláusula OCCURS con la frase DEPENDING ON va seguido de un elemento de datos no subordinado descrito por una cláusula OCCURS .
- Tabla que tiene elementos de longitud variable: un elemento de datos descrito por una cláusula OCCURS contiene un elemento de datos subordinado descrito por una cláusula OCCURS con la frase DEPENDING ON .
- Nombre de índice para una tabla que tiene elementos de longitud variable.
- Elemento de una tabla que tiene elementos de longitud variable.

“Ejemplo: ODO complejo” en la página 74

Tareas relacionadas
“Prevención de errores de índice al cambiar el valor de objeto ODO” en la página 75
“Cómo evitar la superposición al añadir elementos a una tabla de variables” en la página 76

Referencias relacionadas
“Efectos del cambio en el valor del objeto ODO” en la página 75
OCCURS DEPENDING ON cláusula
(*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: ODO complejo

El ejemplo siguiente ilustra los posibles tipos de aparición de ODO complejo.

```
01  FIELD-A.
    02 COUNTER-1                            PIC S99.
    02 COUNTER-2                            PIC S99.
    02 TABLE-1.
       03 RECORD-1 OCCURS 1 TO 5 TIMES
                   DEPENDING ON COUNTER-1    PIC X(3).
    02 EMPLOYEE-NUMBER                      PIC X(5). (1)
    02 TABLE-2 OCCURS 5 TIMES                          (2)(3)
              INDEXED BY INDX.                        (4)
       03 TABLE-ITEM                        PIC 99.   (5)
       03 RECORD-2 OCCURS 1 TO 3 TIMES
                   DEPENDING ON COUNTER-2.
          04 DATA-NUM                       PIC S99.
```

Definición: En el ejemplo, COUNTER-1 es un *objeto ODO*, es decir, es el objeto de la cláusula DEPENDING ON de RECORD-1. Se dice que RECORD-1 es un *asunto ODO*. De forma similar, COUNTER-2 es el objeto ODO del sujeto ODO correspondiente, RECORD-2.

Los tipos de apariciones de ODO complejas que se muestran en el ejemplo anterior son los siguientes:

(1)
Un elemento ubicado de forma variable: EMPLOYEE-NUMBER es un elemento de datos que sigue, pero no está subordinado a, una tabla de longitud variable en el mismo registro de level-01 .

(2)
Una tabla con ubicación variable: TABLE-2 es una tabla que sigue, pero no está subordinada a, una tabla de longitud variable en el mismo registro level-01 .

(3)
Una tabla con elementos de longitud variable: TABLE-2 es una tabla que contiene un elemento de datos subordinado, RECORD-2, cuyo número de apariciones varía en función del contenido de su objeto ODO.

(4)
Un nombre de índice, INDX, para una tabla que tiene elementos de longitud variable.

(5)
Elemento, TABLE-ITEM, de una tabla que tiene elementos de longitud variable.

Cómo se calcula la longitud

La longitud de la parte variable de cada registro es el producto de su objeto ODO y la longitud de su sujeto ODO. Por ejemplo, cuando se hace una referencia a uno de los elementos ODO complejos mostrados anteriormente, la longitud real, si se utiliza, se calcula de la siguiente manera:

- La longitud de TABLE-1 se calcula multiplicando el contenido de COUNTER-1 (el número de apariciones de RECORD-1) por 3 (la longitud de RECORD-1).
- La longitud de TABLE-2 se calcula multiplicando el contenido de COUNTER-2 (el número de apariciones de RECORD-2) por 2 (la longitud de RECORD-2) y añadiendo la longitud de TABLE-ITEM.
- La longitud de FIELD-A se calcula añadiendo las longitudes de COUNTER-1, COUNTER-2, TABLE-1, EMPLOYEE-NUMBERy TABLE-2 veces 5.

Establecimiento de valores de objetos ODO

Debe establecer *cada* objeto ODO en un elemento de grupo antes de hacer referencia a cualquier elemento ODO complejo del grupo. Por ejemplo, antes de hacer referencia a EMPLOYEE-NUMBER en el código anterior, debe establecer COUNTER-1 y COUNTER-2 aunque EMPLOYEE-NUMBER no dependa directamente de ningún objeto ODO para su valor.

Restricción: un objeto ODO no se puede localizar de forma variable.

Efectos del cambio en el valor del objeto ODO

Si un elemento de datos descrito por una cláusula OCCURS con la frase DEPENDING ON va seguido en el mismo grupo por uno o más elementos de datos no subordinados (una forma de ODO complejo), cualquier cambio en el valor del objeto ODO afecta a las referencias posteriores a elementos ODO complejos en el registro.

Por ejemplo:

- El tamaño de cualquier grupo que contenga la cláusula ODO relevante refleja el nuevo valor del objeto ODO.
- Un MOVE a un grupo que contiene el asunto ODO se crea basándose en el nuevo valor del objeto ODO.
- La ubicación de los elementos no subordinados que siguen al elemento descrito con la cláusula ODO se ve afectada por el nuevo valor del objeto ODO. (Para conservar el contenido de los elementos no subordinados, muévalos a un área de trabajo antes de que cambie el valor del objeto ODO y, a continuación, muévalos de nuevo.)

El valor de un objeto ODO puede cambiar cuando mueve datos al objeto ODO o al grupo en el que está contenido. El valor también puede cambiar si el objeto ODO está contenido en un registro que es el destino de una sentencia READ .

Tareas relacionadas
“Prevención de errores de índice al cambiar el valor de objeto ODO” en la página 75
“Cómo evitar la superposición al añadir elementos a una tabla de variables” en la página 76

Prevención de errores de índice al cambiar el valor de objeto ODO

Tenga cuidado si hace referencia a un nombre de índice ODO complejo, es decir, un nombre de índice para una tabla que tiene elementos de longitud variable, después de haber cambiado el valor del objeto ODO para un elemento de datos subordinado en la tabla.

Acerca de esta tarea

Cuando cambia el valor de un objeto ODO, el desplazamiento de bytes en un índice ODO complejo asociado ya no es válido porque la longitud de la tabla ha cambiado. A menos que tome precauciones, tendrá resultados inesperados si luego codifica una referencia al nombre-índice como:

- Una referencia a un elemento de la tabla
- Una sentencia SET con el formato SET *elemento-datos-entero* TO *nombre-índice* (formato 1)
- Una sentencia SET con el formato SET *nombre-índice* UP | DOWN BY *entero* (formato 2)

Para evitar este tipo de error, siga estos pasos:

Procedimiento

1. Guarde el índice en un elemento de datos entero. (Al hacerlo, se produce una conversión implícita: el elemento entero recibe el número de aparición del elemento de tabla que corresponde al desplazamiento en el índice.)
2. Cambie el valor del objeto ODO.
3. Restaure inmediatamente el índice a partir del elemento de datos entero. (Al hacerlo, se produce una conversión implícita: el nombre de índice recibe el desplazamiento que corresponde al número de aparición del elemento de tabla en el elemento entero. El desplazamiento se calcula de acuerdo con la longitud de la tabla y, a continuación, está en vigor.)

Resultados

El código siguiente muestra cómo guardar y restaurar el nombre de índice (que se muestra en “Ejemplo: ODO complejo” en la página 74) cuando el objeto ODO COUNTER-2 cambia.

```
 77  INTEGER-DATA-ITEM-1      PIC 99.
 . . .
     SET INDX TO 5.
*         INDX is valid at this point.
     SET INTEGER-DATA-ITEM-1 TO INDX.
*         INTEGER-DATA-ITEM-1 now has the
*         occurrence number that corresponds to INDX.
     MOVE NEW-VALUE TO COUNTER-2.
*         INDX is not valid at this point.
     SET INDX TO INTEGER-DATA-ITEM-1.
*         INDX is now valid, containing the offset
*         that corresponds to INTEGER-DATA-ITEM-1, and
*         can be used with the expected results.
```

Referencias relacionadas
sentencia SET (*COBOL for Linux en x86 Consulta de lenguaje*)

Cómo evitar la superposición al añadir elementos a una tabla de variables

Tenga cuidado si aumenta el número de elementos en una tabla de aparición de variables que va seguida de uno o más elementos de datos no subordinados en el mismo grupo. Cuando incremente el valor del objeto ODO y añada un elemento a una tabla, puede superponer inadvertidamente los elementos de datos ubicados de forma variable que siguen a la tabla.

Acerca de esta tarea

Para evitar este tipo de error, siga estos pasos:

Procedimiento

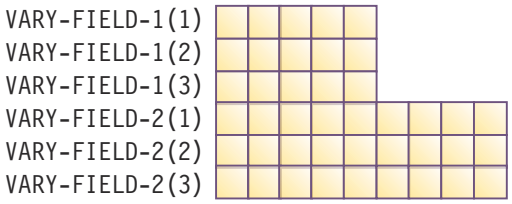
1. Guarde los elementos de datos ubicados de forma variable que siguen a la tabla en otra área de datos.
2. Incremente el valor del objeto ODO.
3. Mueva los datos al nuevo elemento de tabla (si es necesario).
4. Restaure los elementos de datos ubicados de forma variable desde el área de datos donde los ha guardado.

Resultados

En el ejemplo siguiente, suponga que desea añadir un elemento a la tabla VARY-FIELD-1, cuyo número de elementos depende del objeto ODO CONTROL-1. VARY-FIELD-1 va seguido del elemento de datos de ubicación variable no subordinado GROUP-ITEM-1, cuyos elementos podrían estar superpuestos.

```
WORKING-STORAGE SECTION.
01  VARIABLE-REC.
    05  FIELD-1                             PIC X(10).
    05  CONTROL-1                           PIC S99.
    05  CONTROL-2                           PIC S99.
    05  VARY-FIELD-1 OCCURS 1 TO 10 TIMES
          DEPENDING ON CONTROL-1             PIC X(5).
    05  GROUP-ITEM-1.
        10  VARY-FIELD-2
              OCCURS 1 TO 10 TIMES
              DEPENDING ON CONTROL-2         PIC X(9).
01  STORE-VARY-FIELD-2.
    05  GROUP-ITEM-2.
        10  VARY-FLD-2
              OCCURS 1 TO 10 TIMES
              DEPENDING ON CONTROL-2         PIC X(9).
```

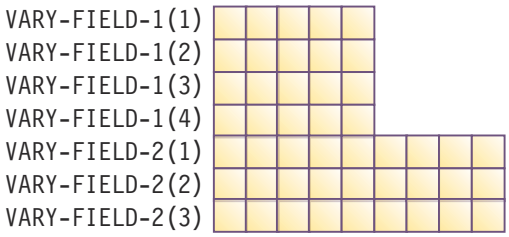
Cada elemento de VARY-FIELD-1 tiene 5 bytes y cada elemento de VARY-FIELD-2 tiene 9 bytes. Si CONTROL-1 y CONTROL-2 contienen ambos el valor 3, puede representar el almacenamiento para VARY-FIELD-1 y VARY-FIELD-2 de la forma siguiente:



Para añadir un cuarto elemento a VARY-FIELD-1, codifique como se indica a continuación para evitar que se superponer los primeros 5 bytes de VARY-FIELD-2. (GROUP-ITEM-2 sirve como almacenamiento temporal para el GROUP-ITEM-1ubicado de forma variable.)

```
MOVE GROUP-ITEM-1 TO GROUP-ITEM-2.
ADD 1 TO CONTROL-1.
MOVE five-byte-field TO
  VARY-FIELD-1 (CONTROL-1).
MOVE GROUP-ITEM-2 TO GROUP-ITEM-1.
```

Puede crear una imagen del almacenamiento actualizado para VARY-FIELD-1 y VARY-FIELD-2 de la siguiente manera:



Tenga en cuenta que el cuarto elemento de VARY-FIELD-1 no se superpuso al primer elemento de VARY-FIELD-2.

Búsqueda de una tabla

COBOL proporciona dos técnicas de búsqueda para tablas: *serie* y *binario*.

Acerca de esta tarea

Para realizar búsquedas en serie, utilice SEARCH e indexación. Para tablas de longitud variable, puede utilizar PERFORM con suscripción o indexación.

Para realizar búsquedas binarias, utilice SEARCH ALL e indexación.

Una búsqueda binaria puede ser considerablemente más eficiente que una búsqueda en serie. Para una búsqueda en serie, el número de comparaciones es del orden de *n*, el número de entradas de la tabla. Para una búsqueda binaria, el número de comparaciones es del orden del logaritmo (base 2) de *n*. Sin embargo, una búsqueda binaria requiere que los elementos de tabla ya estén ordenados.

Tareas relacionadas
“Realizar una búsqueda en serie (SEARCH)” en la página 77
“Realización de una búsqueda binaria (SEARCH ALL)” en la página 79

Realizar una búsqueda en serie (SEARCH)

Utilice la sentencia SEARCH para realizar una búsqueda en serie (secuencial) empezando por el valor de índice actual. Para modificar el valor de índice, utilice la sentencia SET .

Acerca de esta tarea

Las condiciones de la frase WHEN se evalúan en el orden en que aparecen:

- Si no se cumple ninguna de las condiciones, el índice se aumenta para que se corresponda con el siguiente elemento de tabla y las condiciones WHEN se vuelven a evaluar.
- Si se cumple una de las condiciones WHEN , la búsqueda finaliza. El índice permanece apuntando al elemento de tabla que ha satisfecho la condición.
- Si se ha buscado en toda la tabla y no se han cumplido las condiciones, se ejecuta la sentencia imperativa AT END si hay una. Si no ha codificado AT END, el control pasa a la siguiente sentencia del programa.

Sólo puede hacer referencia a un nivel de una tabla (un elemento de tabla) con cada sentencia SEARCH . Para buscar varios niveles de una tabla, utilice sentencias SEARCH anidadas. Delimite cada sentencia SEARCH anidada con END-SEARCH.

Rendimiento: Si la condición encontrada viene después de algún punto intermedio de la tabla, puede acelerar la búsqueda utilizando la sentencia SET para establecer el índice para empezar la búsqueda después de ese punto. Organizar la tabla para que los datos utilizados con más frecuencia estén al principio de la tabla también permite una búsqueda en serie más eficiente. Si la tabla es grande y está preordenada, una búsqueda binaria es más eficaz.

"Ejemplo: búsqueda en serie" en la página 78

Referencias relacionadas
sentencia SEARCH (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: búsqueda en serie

El ejemplo siguiente muestra cómo puede encontrar una serie determinada en la tabla más interna de una tabla tridimensional.

Cada dimensión de la tabla tiene su propio índice (establecido en 1, 4 y 1, respectivamente). La tabla más interna (TABLE-ENTRY3) tiene una clave ascendente.

```
01  TABLE-ONE.
    05 TABLE-ENTRY1 OCCURS 10 TIMES
          INDEXED BY TE1-INDEX.
       10 TABLE-ENTRY2 OCCURS 10 TIMES
             INDEXED BY TE2-INDEX.
          15 TABLE-ENTRY3 OCCURS 5 TIMES
                ASCENDING KEY IS KEY1
                INDEXED BY TE3-INDEX.
             20 KEY1                PIC X(5).
             20 KEY2                PIC X(10).
. . .
PROCEDURE DIVISION.
    . . .
    SET TE1-INDEX TO 1
    SET TE2-INDEX TO 4
    SET TE3-INDEX TO 1
    MOVE "A1234" TO KEY1 (TE1-INDEX, TE2-INDEX, TE3-INDEX + 2)
    MOVE "AAAAAAAA00" TO KEY2 (TE1-INDEX, TE2-INDEX, TE3-INDEX + 2)
    . . .
    SEARCH TABLE-ENTRY3
      AT END
        MOVE 4 TO RETURN-CODE
      WHEN TABLE-ENTRY3(TE1-INDEX, TE2-INDEX, TE3-INDEX)
          = "A1234AAAAAAAA00"
        MOVE 0 TO RETURN-CODE
    END-SEARCH
```

Valores después de la ejecución:

```
TE1-INDEX = 1
TE2-INDEX = 4
```

```
TE3-INDEX points to the TABLE-ENTRY3 item
          that equals "A1234AAAAAAAA00"
RETURN-CODE = 0
```

Realización de una búsqueda binaria (SEARCH ALL)

Si utiliza SEARCH ALL para realizar una búsqueda binaria, no es necesario establecer el índice antes de empezar. El índice es siempre el que está asociado con el primer nombre de índice en la cláusula OCCURS . El índice varía durante la ejecución para maximizar la eficiencia de la búsqueda.

Acerca de esta tarea

Para utilizar la SEARCH ALL sentencia para buscar una tabla, la tabla debe especificar las frases ASCENDING o DESCENDING KEY de la cláusula OCCURS , o ambas, y ya deben estar ordenados en la clave o claves especificadas en las frases ASCENDING y DESCENDING KEY . Puede utilizar una sentencia SORT de formato 2 para ordenar la tabla de acuerdo con sus claves definidas, haciendo así que la tabla se pueda buscar mediante la sentencia SEARCH ALL . Tenga en cuenta que SEARCH ALL devolverá resultados imprevisibles si la tabla no se ha ordenado según las claves.

En la frase WHEN de la sentencia SEARCH ALL , puede probar cualquier clave que se denomine en las frases ASCENDING o DESCENDING KEY para la tabla, pero debe probar todas las claves anteriores, si las hay. La prueba debe ser una condición igual a, y la frase WHEN debe especificar una clave (con el subíndice del primer nombre de índice asociado a la tabla) o un nombre de condición que esté asociado a la clave. La condición WHEN puede ser una condición compuesta formada a partir de condiciones simples que utilizan AND como única conexión lógica.

Cada clave y su objeto de comparación deben ser compatibles según las reglas para la comparación de elementos de datos. Tenga en cuenta, sin embargo, que si una clave se compara con un literal o identificador nacional, la clave debe ser un elemento de datos nacional.

“Ejemplo: búsqueda binaria” en la página 79

Tareas relacionadas
“Definición de una tabla (OCCURS)” en la página 59

Referencias relacionadas
sentencia SEARCH (*COBOL for Linux en x86 Consulta de lenguaje*)
Condiciones de relación generales (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: búsqueda binaria

El ejemplo siguiente muestra cómo puede codificar una búsqueda binaria de una tabla.

Supongamos que define una tabla que contiene 90 elementos de 40 bytes cada uno, y tres claves. Las claves primaria y secundaria (KEY-1 y KEY-2) están en orden ascendente, pero la clave menos significativa (KEY-3) está en orden descendente:

```
01  TABLE-A.
    05 TABLE-ENTRY OCCURS 90 TIMES
           ASCENDING KEY-1, KEY-2
           DESCENDING KEY-3
           INDEXED BY INDX-1.
       10 PART-1       PIC 99.
       10 KEY-1        PIC 9(5).
       10 PART-2       PIC 9(6).
       10 KEY-2        PIC 9(4).
       10 PART-3       PIC 9(18).
       10 KEY-3        PIC 9(5).
```

Puede buscar en esta tabla utilizando las sentencias siguientes:

```
SEARCH ALL TABLE-ENTRY
  AT END
```

```
    PERFORM NOENTRY
  WHEN KEY-1 (INDX-1) = VALUE-1 AND
       KEY-2 (INDX-1) = VALUE-2 AND
       KEY-3 (INDX-1) = VALUE-3
    MOVE PART-1 (INDX-1) TO OUTPUT-AREA
END-SEARCH
```

Si se encuentra una entrada en la que cada una de las tres claves es igual al valor con el que se compara (VALUE-1, VALUE-2y VALUE-3, respectivamente), PART-1 de dicha entrada se mueve a OUTPUT-AREA. Si no se encuentra ninguna clave coincidente en las entradas de TABLE-A, se realiza la rutina NOENTRY .

Ordenación de una tabla

Puede ordenar una tabla utilizando la sentencia SORT de formato 2. Forma parte de Estándar COBOL 2002.

Acerca de esta tarea

La sentencia SORT de formato 2 clasifica los elementos de tabla de acuerdo con las claves de tabla especificadas, y es especialmente útil para las tablas utilizadas con SEARCH ALL. Puede especificar las claves para ordenar como parte de la definición de tabla, que también se puede utilizar en la sentencia SEARCH ALL . De forma alternativa, también puede especificar las claves para ordenar como parte de la sentencia SORT , ya sea si desea ordenar la tabla utilizando claves diferentes a las especificadas en la definición de tabla, o si la tabla no tiene claves especificadas.

Con la sentencia SORT de formato 2, no es necesario utilizar los procedimientos de entrada y salida como lo hace con la sentencia SORT de formato 1.

Consulte el ejemplo siguiente en el que la tabla se ordena basándose en las claves especificadas:

```
WORKING-STORAGE SECTION.
01 GROUP-ITEM.
    05 TABL OCCURS 10 TIMES
       10 ELEM-ITEM1 PIC X.
       10 ELEM-ITEM2 PIC X.
       10 ELEM-ITEM3 PIC X.
...
PROCEDURE DIVISION.
    ...
    SORT TABL DESCENDING ELEM-ITEM2 ELEM-ITEM3.
    IF TABL (1)...
```

Referencias relacionadas
Sentencia SORT (*COBOL for Linux en x86 Consulta de lenguaje*)
“Utilización de la sentencia SORT de formato 2 para ordenar una tabla” en la página 547

Proceso de elementos de tabla utilizando funciones intrínsecas

Puede utilizar funciones intrínsecas para procesar elementos de tabla alfabéticos, alfanuméricos, nacionales o numéricos. (Sólo puede procesar elementos de datos DBCS con la función intrínseca NATIONAL-OF .) Las descripciones de datos de los elementos de tabla deben ser compatibles con los requisitos para los argumentos de función.

Acerca de esta tarea

Utilice un subíndice o índice para hacer referencia a un elemento de datos individual como argumento de función. Por ejemplo, suponiendo que Table-One es una matriz de 3 x 3 elementos numéricos, puede encontrar la raíz cuadrada del elemento central utilizando esta sentencia:

```
Compute X = Function Sqrt(Table-One(2,2))
```

A menudo, es posible que tenga que procesar de forma iterativa los datos en tablas. Para las funciones intrínsecas que aceptan varios argumentos, puede utilizar el subíndice ALL para hacer referencia a todos los elementos de la tabla o en una sola dimensión de la tabla. La iteración se maneja automáticamente, lo que puede hacer que el código sea más corto y sencillo.

Puede mezclar escalares y argumentos de matriz para funciones que aceptan varios argumentos:

```
Compute Table-Median = Function Median(Arg1 Table-One(ALL))
```

"Ejemplo: proceso de tablas utilizando funciones intrínsecas" en la página 81

Tareas relacionadas
"Utilización de funciones intrínsecas (funciones incorporadas)" en la página 33
"Conversión de elementos de datos (funciones intrínsecas)" en la página 106
"Evaluación de elementos de datos (funciones intrínsecas)" en la página 109

Referencias relacionadas
Funciones intrínsecas (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: proceso de tablas utilizando funciones intrínsecas

Estos ejemplos muestran cómo puede aplicar una función intrínseca a algunos o a todos los elementos de una tabla utilizando el subíndice ALL .

Suponiendo que Table-Two es una matriz 2 x 3 x 2, la sentencia siguiente añade los valores en los elementos Table-Two(1,3,1), Table-Two(1,3,2), Table-Two(2,3,1)y Table-Two(2,3,2):

```
Compute Table-Sum = FUNCTION SUM (Table-Two(ALL, 3, ALL))
```

El ejemplo siguiente calcula varios valores de salario para todos los empleados cuyos salarios están codificados en Employee-Table:

```
01  Employee-Table.
    05 Emp-Count      Pic s9(4) usage binary.
    05 Emp-Record     Occurs 1 to 500 times
                        depending on Emp-Count.
       10 Emp-Name    Pic x(20).
       10 Emp-Idme    Pic 9(9).
       10 Emp-Salary  Pic 9(7)v99.
. . .
Procedure Division.
    Compute Max-Salary    = Function Max(Emp-Salary(ALL))
    Compute I             = Function Ord-Max(Emp-Salary(ALL))
    Compute Avg-Salary    = Function Mean(Emp-Salary(ALL))
    Compute Salary-Range  = Function Range(Emp-Salary(ALL))
    Compute Total-Payroll = Function Sum(Emp-Salary(ALL))
```

Capítulo 5. Selección y repetición de acciones de programa

Acerca de esta tarea

Utilice el lenguaje de control COBOL para elegir acciones de programa basadas en el resultado de las pruebas lógicas, para iterar por partes seleccionadas del programa y los datos, y para identificar las sentencias que se deben realizar como un grupo.

Estos controles incluyen las sentencias IF, EVALUATEy PERFORM , y el uso de conmutadores y distintivos.

Tareas relacionadas
“Selección de acciones de programa” en la página 83
“Acciones de programa repetitivas” en la página 91

Selección de acciones de programa

Puede proporcionar distintas acciones de programa en función del valor probado de uno o varios elementos de datos.

Acerca de esta tarea

Las sentencias IF y EVALUATE en COBOL prueban uno o más elementos de datos mediante una expresión condicional.

Tareas relacionadas
“Codificación de una opción de acciones” en la página 83
“Codificación de expresiones condicionales” en la página 87

Referencias relacionadas
sentencia IF (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia EVALUATE (*COBOL for Linux en x86 Consulta de lenguaje*)

Codificación de una opción de acciones

Utilice IF . . . ELSE para codificar una opción entre dos acciones de proceso. (La palabra THEN es opcional.) Utilice la sentencia EVALUATE para codificar una opción entre tres o más acciones posibles.

Acerca de esta tarea

```
IF condition-p
  statement-1
ELSE
  statement-2
END-IF
```

Cuando una de las dos opciones de proceso no es ninguna acción, codifique la sentencia IF con o sin ELSE. Puesto que la cláusula ELSE es opcional, puede codificar la sentencia IF como se indica a continuación:

```
IF condition-q
  statement-1
END-IF
```

Esta codificación es adecuada para casos simples. Para la lógica compleja, es probable que tenga que utilizar la cláusula ELSE . Por ejemplo, supongamos que tiene sentencias IF anidadas en las que sólo hay una acción para una de las opciones de proceso. Puede utilizar la cláusula ELSE y codificar la rama nula de la sentencia IF con la sentencia CONTINUE :

```
IF condition-q
  statement-1
ELSE
  CONTINUE
END-IF
```

La sentencia EVALUATE es un formato expandido de la sentencia IF que le permite evitar la anidación de sentencias IF , un origen común de errores lógicos y la depuración de problemas.

Tareas relacionadas
“Utilización de sentencias IF anidadas” en la página 84
“Utilización de la sentencia EVALUATE” en la página 85
“Codificación de expresiones condicionales” en la página 87

Utilización de sentencias IF anidadas

Si una sentencia IF contiene una sentencia IF como una de sus posibles ramas, se dice que las sentencias IF están *anidadas*. Teóricamente, no hay ningún límite en la profundidad de las sentencias IF anidadas.

Acerca de esta tarea

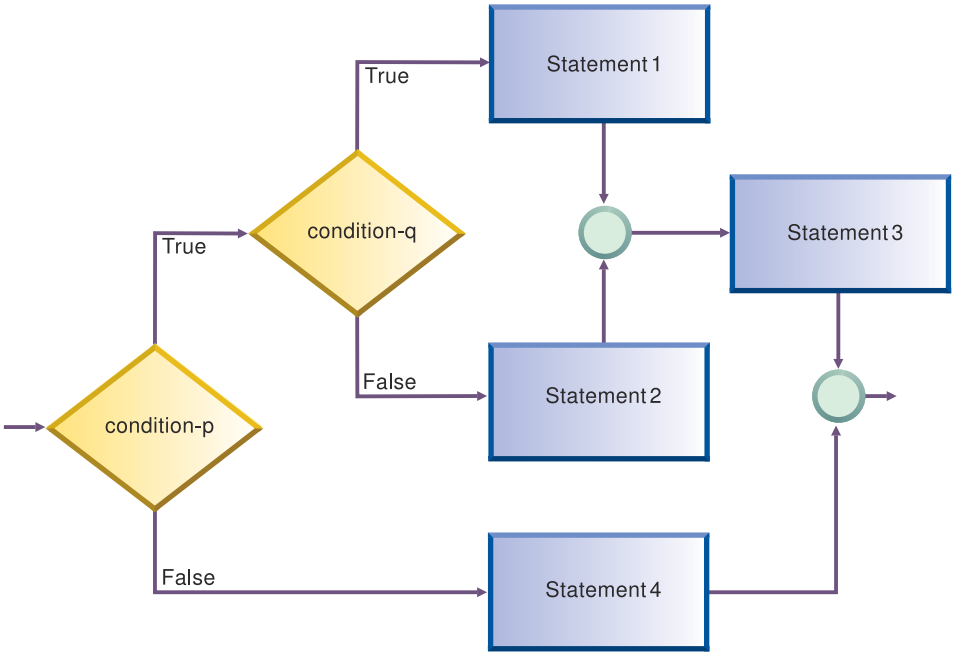
Sin embargo, utilice las sentencias IF anidadas con moderación. La lógica puede ser difícil de seguir, aunque los terminadores de ámbito explícitos y la sangría pueden ayudar. Si un programa tiene que probar una variable para más de dos valores, EVALUATE es probablemente una opción mejor.

El siguiente pseudocódigo muestra una sentencia IF anidada:

```
IF condition-p
  IF condition-q
    statement-1
  ELSE
    statement-2
  END-IF
  statement-3
ELSE
  statement-4
END-IF
```

En el pseudocódigo anterior, una sentencia IF y una estructura secuencial se anidan en una rama del IFexterno. En esta estructura, el END-IF que cierra el IF anidado es muy importante. Utilice END-IF en lugar de un punto, porque un punto finalizaría también la estructura externa de IF .

La figura siguiente muestra la estructura lógica del pseudocódigo anterior.



Tareas relacionadas
"Codificación de una opción de acciones" en la página 83

Referencias relacionadas
Terminadores de ámbito explícitos (*COBOL for Linux en x86 Consulta de lenguaje*)

Utilización de la sentencia EVALUATE

Puede utilizar la sentencia EVALUATE en lugar de una serie de IF sentencias anidadas para probar varias condiciones y especificar una acción diferente para cada una. Por lo tanto, puede utilizar la sentencia EVALUATE para implementar una *estructura de caso* o tabla de decisiones.

Acerca de esta tarea

También puede utilizar la sentencia EVALUATE para provocar que varias condiciones conduzcan al mismo proceso, tal como se muestra en estos ejemplos:

"Ejemplo: EVALUATE utilizando frase THRU" en la página 86
"Ejemplo: EVALUATE utilizando varias frases WHEN" en la página 86

En una sentencia EVALUATE , los operandos antes de la frase WHEN se denominan *sujetos de selección*y los operandos de la frase WHEN se denominan *objetos de selección*. Los sujetos de selección pueden ser identificadores, literales, expresiones condicionales o la palabra TRUE o FALSE. Los objetos de selección pueden ser identificadores, literales, expresiones condicionales o aritméticas, o la palabra TRUE, FALSEo ANY.

Puede separar varios asuntos de selección con la frase ALSO . Puede separar varios objetos de selección con la frase ALSO . El número de objetos de selección dentro de cada conjunto de objetos de selección debe ser igual al número de sujetos de selección, tal como se muestra en este ejemplo:

"Ejemplo: EVALUATE probando varias condiciones" en la página 87

Los identificadores, literales o expresiones aritméticas que aparecen dentro de un objeto de selección deben ser operandos válidos para compararlos con el operando correspondiente en el conjunto de sujetos de selección. Las condiciones o la palabra TRUE o FALSE que aparecen en un objeto de selección deben corresponder a una expresión condicional o a la palabra TRUE o FALSE en el conjunto de sujetos

de selección. (Puede utilizar la palabra ANY como objeto de selección para corresponder a cualquier tipo de asunto de selección.)

La ejecución de la sentencia EVALUATE finaliza cuando se produce una de las condiciones siguientes:

- Se realizan las sentencias asociadas con la frase WHEN seleccionada.
- Se realizan las sentencias asociadas con la frase WHEN OTHER .
- No se cumplen las condiciones de WHEN .

Las frases WHEN se prueban en el orden en que aparecen en el programa fuente. Por lo tanto, debe ordenar estas frases para obtener el mejor rendimiento. En primer lugar, codifique la frase WHEN que contiene objetos de selección que es más probable que se satisfagan, luego la siguiente más probable, y así sucesivamente. Una excepción es la frase WHEN OTHER , que debe ser la última.

Tareas relacionadas
"Codificación de una opción de acciones" en la página 83

Referencias relacionadas
sentencia EVALUATE (*COBOL for Linux en x86 Consulta de lenguaje*)
Condiciones de relación generales (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: EVALUATE utilizando frase THRU

Este ejemplo muestra cómo puede codificar varias condiciones en un rango de valores para llevar a la misma acción de proceso codificando la frase THRU . Los operandos de una frase THRU deben ser de la misma clase.

En este ejemplo, CARPOOL-SIZE es el *asunto de selección*; 1, 2y 3 THRU 6 son los *objetos de selección*:

```
EVALUATE CARPOOL-SIZE
  WHEN 1
    MOVE "SINGLE" TO PRINT-CARPOOL-STATUS
  WHEN 2
    MOVE "COUPLE" TO PRINT-CARPOOL-STATUS
  WHEN 3 THRU 6
    MOVE "SMALL GROUP" TO PRINT-CARPOOL STATUS
  WHEN OTHER
    MOVE "BIG GROUP" TO PRINT-CARPOOL STATUS
END-EVALUATE
```

Las siguientes sentencias IF anidadas representan la misma lógica:

```
IF CARPOOL-SIZE = 1 THEN
  MOVE "SINGLE" TO PRINT-CARPOOL-STATUS
ELSE
  IF CARPOOL-SIZE = 2 THEN
    MOVE "COUPLE" TO PRINT-CARPOOL-STATUS
  ELSE
    IF CARPOOL-SIZE >= 3 and CARPOOL-SIZE <= 6 THEN
      MOVE "SMALL GROUP" TO PRINT-CARPOOL-STATUS
    ELSE
      MOVE "BIG GROUP" TO PRINT-CARPOOL-STATUS
    END-IF
  END-IF
END-IF
```

Ejemplo: EVALUATE utilizando varias frases WHEN

El ejemplo siguiente muestra que puede codificar varias frases WHEN si varias condiciones deben llevar a la misma acción. Hacerlo le proporciona más flexibilidad que utilizar solo la frase THRU , porque las condiciones no tienen que evaluar los valores en un rango ni tienen la misma clase.

```
EVALUATE MARITAL-CODE
  WHEN "M"
    ADD 2 TO PEOPLE-COUNT
  WHEN "S"
  WHEN "D"
```

```
  WHEN "W"
    ADD 1 TO PEOPLE-COUNT
END-EVALUATE
```

Las siguientes sentencias IF anidadas representan la misma lógica:

```
IF MARITAL-CODE = "M" THEN
  ADD 2 TO PEOPLE-COUNT
ELSE
  IF MARITAL-CODE = "S" OR
     MARITAL-CODE = "D" OR
     MARITAL-CODE = "W" THEN
       ADD 1 TO PEOPLE-COUNT
  END-IF
END-IF
```

Ejemplo: EVALUATE probando varias condiciones

Este ejemplo muestra el uso de la frase ALSO para separar dos sujetos de selección (`True ALSO True`) y para separar los dos objetos de selección correspondientes dentro de cada conjunto de objetos de selección (por ejemplo, `When A + B < 10 Also C = 10`).

Ambos objetos de selección en una frase WHEN deben satisfacer la condición TRUE, TRUE antes de que se realice la acción asociada. Si ambos objetos no se evalúan como TRUE, se procesa la frase WHEN siguiente.

```
Identification Division.
  Program-ID. MiniEval.
Environment Division.
  Configuration Section.
Data Division.
  Working-Storage Section.
  01   Age             Pic  999.
  01   Sex             Pic  X.
  01   Description     Pic  X(15).
  01   A               Pic  999.
  01   B               Pic  9999.
  01   C               Pic  9999.
  01   D               Pic  9999.
  01   E               Pic  99999.
  01   F               Pic  999999.
Procedure Division.
  PN01.
    Evaluate True Also True
      When Age < 13 Also Sex = "M"
        Move "Young Boy" To Description
      When Age < 13 Also Sex = "F"
        Move "Young Girl" To Description
      When Age > 12 And Age < 20 Also Sex = "M"
        Move "Teenage Boy" To Description
      When Age > 12 And Age < 20 Also Sex = "F"
        Move "Teenage Girl" To Description
      When Age > 19 Also Sex = "M"
        Move "Adult Man" To Description
      When Age > 19 Also Sex = "F"
        Move "Adult Woman" To Description
      When Other
        Move "Invalid Data" To Description
    End-Evaluate
    Evaluate True Also True
      When A + B < 10 Also C = 10
        Move "Case 1" To Description
      When A + B > 50 Also C = ( D + E ) / F
        Move "Case 2" To Description
      When Other
        Move "Case Other" To Description
    End-Evaluate
    Stop Run.
```

Codificación de expresiones condicionales

Utilizando las sentencias IF y EVALUATE , puede codificar acciones de programa que se realizarán en función del valor de verdad de una expresión condicional.

Acerca de esta tarea

Puede especificar las condiciones siguientes:

- Condiciones de relación, como por ejemplo:
  - Comparaciones numéricas
  - Comparaciones alfanuméricas
  - Comparaciones DBCS
  - Comparaciones nacionales
- Condiciones de clase; por ejemplo, para probar si un elemento de datos:
  - `IS NUMERIC`
  - `IS ALPHABETIC`
  - `IS ALPHABETIC-LOWER`
  - `IS ALPHABETIC-UPPER`
  - `IS DBCS`
  - `IS KANJI`
- Condiciones de nombre de condición, para probar el valor de una variable condicional que defina
- Firmar condiciones, para probar si un operando numérico `IS POSITIVE`, `NEGATIVE`o `ZERO`
- Condiciones de estado de conmutador, para probar el estado de los conmutadores UPSI que ha especificado en el párrafo `SPECIAL-NAMES`
- Condiciones complejas, como:
  - Condiciones negadas; por ejemplo, `NOT (A IS EQUAL TO B)`
  - Condiciones combinadas (condiciones combinadas con operadores lógicos `AND` o `OR`)

Conceptos relacionados
“Conmutadores y distintivos” en la página 88

Tareas relacionadas
“Definición de conmutadores y distintivos” en la página 89
“Restablecimiento de conmutadores y distintivos” en la página 90
“Comprobación de datos incompatibles (prueba de clase numérica)” en la página 48
“Comparación de datos nacionales (UTF-16)” en la página 205
“Prueba de caracteres DBCS válidos” en la página 212

Referencias relacionadas
“UPSI” en la página 318
Condiciones de relación generales (*COBOL for Linux en x86 Consulta de lenguaje*)
Condición de clase (*COBOL for Linux en x86 Consulta de lenguaje*)
Reglas para entradas de nombre de condición (*COBOL for Linux en x86 Consulta de lenguaje*)
Condición de firma (*COBOL for Linux en x86 Consulta de lenguaje*)
Condiciones combinadas (*COBOL for Linux en x86 Consulta de lenguaje*)

Conmutadores y distintivos

Algunas decisiones de programa se basan en si el valor de un elemento de datos es verdadero o falso, activado o desactivado, sí o no. Controle estas decisiones bidireccionales utilizando elementos level-88 con nombres significativos (*condition-names*) para actuar como conmutadores.

Otras decisiones de programa dependen del valor concreto o del rango de valores de un elemento de datos. Cuando se utilizan nombres de condición para proporcionar más que valores de encendido o apagado a un campo, el campo se suele denominar *distintivo*.

Los distintivos y conmutadores hacen que el código sea más fácil de cambiar. Si necesita cambiar los valores de una condición, sólo tiene que cambiar el valor de ese nombre de condición level-88 .

Por ejemplo, supongamos que un programa utiliza un nombre de condición para probar un campo para un rango de salario determinado. Si el programa debe cambiarse para comprobar un rango de salarios diferente, sólo tiene que cambiar el valor del nombre-condición en DATA DIVISION. No es necesario realizar cambios en PROCEDURE DIVISION.

Tareas relacionadas
“Definición de conmutadores y distintivos” en la página 89
“Restablecimiento de conmutadores y distintivos” en la página 90

Definición de conmutadores y distintivos

En DATA DIVISION, defina elementos de level-88 que actuarán como conmutadores o distintivos y deles nombres significativos.

Acerca de esta tarea

Para probar más de dos valores con distintivos, asigne más de un nombre de condición a un campo utilizando varios elementos level-88 .

El lector puede seguir fácilmente el código si elige nombres de condición significativos y si los valores asignados a ellos tienen alguna asociación con valores lógicos.

“Ejemplo: conmutadores” en la página 89
“Ejemplo: distintivos” en la página 89

Ejemplo: conmutadores

Los ejemplos siguientes muestran cómo puede utilizar elementos de level-88 para probar diversas condiciones de valor binario (desactivado) en el programa.

Por ejemplo, para probar la condición de fin de archivo para un archivo de entrada denominado Transaction-File, puede utilizar las siguientes definiciones de datos:

```
WORKING-STORAGE SECTION.
01  Switches.
    05  Transaction-EOF-Switch  Pic X  value space.
        88  Transaction-EOF     value "y".
```

La descripción level-88 indica que una condición denominada Transaction-EOF está en vigor cuando Transaction-EOF-Switch tiene el valor 'y'. La referencia a Transaction-EOF en PROCEDURE DIVISION expresa la misma condición que la prueba Transaction-EOF-Switch = "y". Por ejemplo, la sentencia siguiente hace que se imprima un informe sólo si Transaction-EOF-Switch se ha establecido en 'y':

```
If Transaction-EOF Then
    Perform Print-Report-Summary-Lines
End-if
```

Ejemplo: distintivos

Los ejemplos siguientes muestran cómo puede utilizar varios elementos level-88 junto con una sentencia EVALUATE para determinar cuál de varias condiciones de un programa es verdadera.

Considere, por ejemplo, un programa que actualiza un archivo main . Las actualizaciones se leen desde un archivo de transacción. Los registros del archivo contienen un campo que indica cuál de las tres funciones se va a realizar: añadir, cambiar o suprimir. En la descripción de registro del archivo de entrada, codifique un campo para el código de función utilizando elementos level-88 :

```
01  Transaction-Input Record
    05  Transaction-Type         Pic X.
```

```
        88  Add-Transaction      Value "A".
        88  Change-Transaction   Value "C".
        88  Delete-Transaction   Value "D".
```

El código en PROCEDURE DIVISION para probar estos nombres de condición para determinar qué función se va a realizar podría ser similar al siguiente:

```
Evaluate True
  When Add-Transaction
    Perform Add-Main-Record-Paragraph
  When Change-Transaction
    Perform Update-Existing-Record-Paragraph
  When Delete-Transaction
    Perform Delete-Main-Record-Paragraph
End-Evaluate
```

Restablecimiento de conmutadores y distintivos

En todo el programa, es posible que tenga que restablecer los conmutadores o distintivos a los valores originales que tenían en sus descripciones de datos. Para ello, utilice una sentencia SET o defina un elemento de datos para pasar al conmutador o distintivo.

Acerca de esta tarea

Cuando se utiliza la sentencia SET *nombre-condición* TO TRUE , el conmutador o distintivo se establece en el valor original que se le asignó en su descripción de datos. Para un elemento de level-88 que tiene varios valores, SET *nombre-condición* TO TRUE asigna el primer valor (A en el ejemplo siguiente):

```
88 Record-is-Active Value "A" "O" "S"
```

El uso de la sentencia SET y de nombres de condición significativos facilita a los lectores el seguimiento del código.

“Ejemplo: activar conmutador” en la página 90
“Ejemplo: desactivar desactivación” en la página 91

Ejemplo: activar conmutador

Los ejemplos siguientes muestran cómo puede activar un conmutador codificando una sentencia SET que mueve el valor de nombre de condición a la variable condicional.

Por ejemplo, la sentencia SET del ejemplo siguiente tiene el mismo efecto que codificar la sentencia `Move "y" to Transaction-EOF-Switch`:

```
01  Switches
    05  Transaction-EOF-Switch   Pic X  Value space.
        88  Transaction-EOF             Value "y".
. . .
Procedure Division.
000-Do-Main-Logic.
    Perform 100-Initialize-Paragraph
    Read Update-Transaction-File
      At End Set Transaction-EOF to True
    End-Read
```

El ejemplo siguiente muestra cómo asignar un valor a un campo en un registro de salida basado en el código de transacción de un registro de entrada:

```
01  Input-Record.
    05  Transaction-Type         Pic X(9).
01  Data-Record-Out.
    05  Data-Record-Type         Pic X.
        88 Record-Is-Active          Value "A".
```

```
        88 Record-Is-Suspended       Value "S".
        88 Record-Is-Deleted         Value "D".
    05  Key-Field                Pic X(5).
. . .
Procedure Division.
    Evaluate Transaction-Type of Input-Record
      When "ACTIVE"
        Set Record-Is-Active to TRUE
      When "SUSPENDED"
        Set Record-Is-Suspended to TRUE
      When "DELETED"
        Set Record-Is-Deleted to TRUE
    End-Evaluate
```

Ejemplo: desactivar desactivación

El ejemplo siguiente muestra cómo puede desactivar un conmutador codificando una sentencia MOVE que mueve el valor de nombre de condición a la variable condicional.

Por ejemplo, puede utilizar un elemento de datos denominado SWITCH-OFF para establecer un conmutador de activación/desactivación, como en el código siguiente, que restablece un conmutador para indicar que no se ha alcanzado el final del archivo:

```
01  Switches
    05  Transaction-EOF-Switch       Pic X  Value space.
        88  Transaction-EOF                 Value "y".
01  SWITCH-OFF                      Pic X  Value "n".
. . .
Procedure Division.
    . . .
    Move SWITCH-OFF to Transaction-EOF-Switch
```

Acciones de programa repetitivas

Utilice una sentencia PERFORM para repetir el mismo código (es decir, bucle) un número especificado de veces o basándose en el resultado de una decisión.

Acerca de esta tarea

También puede utilizar una sentencia PERFORM para ejecutar un párrafo y, a continuación, devolver implícitamente el control a la siguiente sentencia ejecutable. En efecto, esta sentencia PERFORM es una forma de codificar una subrutina cerrada que puede entrar desde muchas partes diferentes del programa.

Las sentencias PERFORM pueden estar en línea o fuera de línea.

Tareas relacionadas
“Elección de PERFORM en línea o fuera de línea” en la página 91
“Codificación de un bucle” en la página 92
“Bucle a través de una tabla” en la página 93
“Ejecución de varios párrafos o secciones” en la página 94

Referencias relacionadas
sentencia PERFORM (*COBOL for Linux en x86 Consulta de lenguaje*)

Elección de PERFORM en línea o fuera de línea

Una PERFORM en línea es una sentencia imperativa que se ejecuta en el flujo normal de un programa; una PERFORM fuera de línea implica una ramificación a un párrafo con nombre y un retorno implícito de dicho párrafo.

Acerca de esta tarea

Para determinar si se debe codificar una sentencia PERFORM en línea o fuera de línea, responda a las preguntas siguientes:

- ¿Se utiliza la sentencia PERFORM en varios lugares?

  Utilice un PERFORM fuera de línea cuando desee utilizar la misma parte de código en varios lugares del programa.
- ¿Qué colocación de la declaración será más fácil de leer?

  Si el código que se va a realizar es corto, un PERFORM en línea puede ser más fácil de leer. Pero si el código se extiende a través de varias pantallas, el flujo lógico del programa podría ser más claro si utiliza un PERFORMfuera de línea. (Sin embargo, cada párrafo de la programación estructurada debe realizar una función lógica.)
- ¿Cuáles son las compensaciones de eficiencia?

  Un PERFORM en línea evita la sobrecarga de ramificación que se produce con un PERFORMfuera de línea. Pero incluso la codificación PERFORM fuera de línea puede mejorar la optimización de código, por lo que las mejoras de eficiencia no se deben sobreenfatizar.

En el estándar COBOL de 1974, la sentencia PERFORM está fuera de línea y, por lo tanto, requiere una ramificación para un procedimiento independiente y un retorno implícito. Si el procedimiento realizado está en el flujo secuencial subsiguiente del programa, también se ejecuta en ese flujo lógico. Para evitar esta ejecución adicional, coloque el procedimiento fuera del flujo secuencial normal (por ejemplo, después de GOBACK) o codifique una ramificación alrededor de él.

El asunto de un PERFORM en línea es una sentencia imperativa. Por lo tanto, debe codificar sentencias (que no sean sentencias imperativas) dentro de un PERFORM en línea con terminadores de ámbito explícitos.

“Ejemplo: sentencia PERFORM en línea” en la página 92

Ejemplo: sentencia PERFORM en línea

Este ejemplo muestra la estructura de una sentencia PERFORM en línea que tiene los terminadores de ámbito necesarios y la frase END-PERFORM necesaria.

```
   Perform 100-Initialize-Paragraph
* The following statement is an inline PERFORM:
   Perform Until Transaction-EOF
      Read Update-Transaction-File Into WS-Transaction-Record
         At End
            Set Transaction-EOF To True
         Not At End
            Perform 200-Edit-Update-Transaction
            If No-Errors
               Perform 300-Update-Commuter-Record
            Else
               Perform 400-Print-Transaction-Errors
* End-If is a required scope terminator
            End-If
            Perform 410-Re-Initialize-Fields
* End-Read is a required scope terminator
      End-Read
   End-Perform
```

Codificación de un bucle

Utilice la sentencia PERFORM . . . TIMES para ejecutar un procedimiento un número especificado de veces.

Acerca de esta tarea

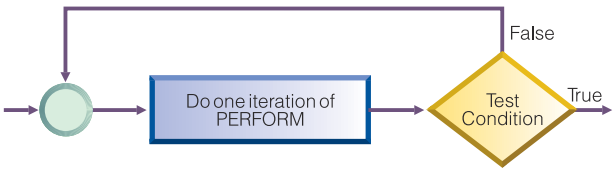
```
PERFORM 010-PROCESS-ONE-MONTH 12 TIMES
INSPECT . . .
```

En el ejemplo anterior, cuando el control alcanza la sentencia PERFORM , el código del procedimiento 010-PROCESS-ONE-MONTH se ejecuta 12 veces antes de que el control se transfiera a la sentencia INSPECT .

Utilice la sentencia PERFORM . . . UNTIL para ejecutar un procedimiento hasta que se cumpla una condición que elija. Puede utilizar cualquiera de los formatos siguientes:

```
PERFORM . . . WITH TEST AFTER  . . . . UNTIL . . .
PERFORM . . . [WITH TEST BEFORE] . . . UNTIL . . .
```

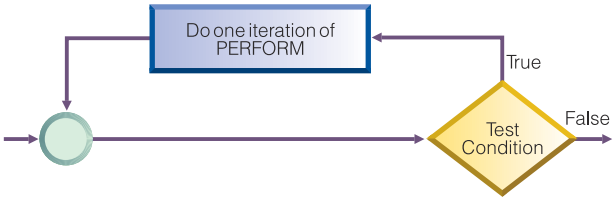
Utilice la sentencia PERFORM . . . WITH TEST AFTER . . . UNTIL si desea ejecutar el procedimiento al menos una vez y probar antes de cualquier ejecución posterior. Esta sentencia es equivalente a una estructura do-until:



En el ejemplo siguiente, la frase WITH TEST BEFORE implícita proporciona una estructura do-while:

```
PERFORM  010-PROCESS-ONE-MONTH
  UNTIL MONTH GREATER THAN 12
INSPECT . . .
```

Cuando el control alcanza la sentencia PERFORM , se prueba la condición MONTH GREATER THAN 12 . Si se cumple la condición, el control se transfiere a la sentencia INSPECT . Si no se cumple la condición, se ejecuta 010-PROCESS-ONE-MONTH y se vuelve a probar la condición. Este ciclo continúa hasta que la condición se prueba como verdadera. (Para que el programa sea más fácil de leer, es posible que desee codificar la cláusula WITH TEST BEFORE .)



Bucle a través de una tabla

Puede utilizar la sentencia PERFORM . . . VARYING para inicializar una tabla. En este formato de la sentencia PERFORM , una variable se aumenta o disminuye y se prueba hasta que se satisface una condición.

Acerca de esta tarea

Por lo tanto, utilice la sentencia PERFORM para controlar el bucle a través de una tabla. Puede utilizar cualquiera de estos formatos:

```
PERFORM . . . WITH TEST AFTER  . . . . VARYING . . . UNTIL . . .
PERFORM . . . [WITH TEST BEFORE] . . . VARYING . . . UNTIL . . .
```

La siguiente sección de código muestra un ejemplo de bucle a través de una tabla para comprobar si hay datos no válidos:

```
PERFORM TEST AFTER VARYING WS-DATA-IX
      FROM 1 BY 1 UNTIL WS-DATA-IX = 12
   IF WS-DATA (WS-DATA-IX) EQUALS SPACES
```

```
      SET SERIOUS-ERROR TO TRUE
      DISPLAY ELEMENT-NUM-MSG5
   END-IF
END-PERFORM
INSPECT . . .
```

Cuando el control alcanza la sentencia PERFORM anterior, WS-DATA-IX se establece en 1 y se ejecuta la sentencia PERFORM . A continuación, se prueba la condición WS-DATA-IX = 12 . Si la condición es verdadera, el control pasa a la sentencia INSPECT . Si la condición es falsa, WS-DATA-IX se incrementa en 1, se ejecuta la sentencia PERFORM y se vuelve a probar la condición. Este ciclo de ejecución y pruebas continúa hasta que WS-DATA-IX es igual a 12.

El bucle anterior controla la comprobación de entrada para los 12 campos del elemento WS-DATA. Los campos vacíos no están permitidos en la aplicación, por lo que la sección de bucles de código y emite mensajes de error según corresponda.

Ejecución de varios párrafos o secciones

En la programación estructurada, normalmente ejecuta un único párrafo. Sin embargo, puede ejecutar un grupo de párrafos, o una sola sección o grupo de secciones, codificando la sentencia PERFORM . . . THRU .

Acerca de esta tarea

Cuando utilice la sentencia PERFORM . . . THRU , codifique un párrafo-sentenciaEXIT para indicar claramente el punto final de una serie de párrafos.

Tareas relacionadas
“Proceso de elementos de tabla utilizando funciones intrínsecas” en la página 80

Referencias relacionadas
sentencia EXIT PERFORM o EXIT PERFORM CYCLE
(*COBOL for Linux en x86 Consulta de lenguaje*)
EXIT PÁRRAFO o EXIT SECTION
(*COBOL for Linux en x86 Consulta de lenguaje*)

Capítulo 6. Manejo de series

Acerca de esta tarea

COBOL proporciona construcciones de lenguaje para realizar muchas operaciones diferentes en elementos de datos de serie.

Por ejemplo, puede:

- Unir o dividir elementos de datos.
- Manipule las series terminadas en nulo, como por ejemplo contar o mover caracteres.
- Consulte las subseries por su posición ordinal y, si es necesario, por su longitud.
- Contabilizar y sustituir elementos de datos, como contar el número de veces que aparece un carácter específico en un elemento de datos.
- Convertir elementos de datos como, por ejemplo, cambiar a mayúsculas o minúsculas.
- Evalúe los elementos de datos como, por ejemplo, determine la longitud de un elemento de datos.

Tareas relacionadas
“Unión de elementos de datos (STRING)” en la página 95
“División de elementos de datos (UNSTRING)” en la página 97
“Manipulación de series terminadas en nulo” en la página 100
“Referencia a subseries de elementos de datos” en la página 101
“Recuento y sustitución de elementos de datos (INSPECT)” en la página 104
“Conversión de elementos de datos (funciones intrínsecas)” en la página 106
“Evaluación de elementos de datos (funciones intrínsecas)” en la página 109
Capítulo 10, “Tratamiento de datos en un entorno internacional”, en la página 189

Unión de elementos de datos (STRING)

Utilice la sentencia STRING para unir todos o partes de varios elementos de datos o literales en un elemento de datos. Una sentencia STRING puede ocupar el lugar de varias sentencias MOVE .

Acerca de esta tarea

La sentencia STRING transfiere datos a un elemento de datos de recepción en el orden que indique. En la sentencia STRING también especifica:

- Un delimitador para cada conjunto de campos de envío que, si se encuentra, hace que los campos de envío dejen de transferirse (fraseDELIMITED BY)
- (Opcional) Acción que se debe realizar si el campo de recepción se rellena antes de que se hayan procesado todos los datos de envío (fraseON OVERFLOW)
- (Opcional) Un elemento de datos entero que indica la posición de carácter más a la izquierda dentro del campo de recepción al que se deben transferir los datos (fraseWITH POINTER)

El elemento de datos de recepción no debe ser un elemento editado, o un elemento de visualización o de coma flotante nacional. Si el elemento de datos de recepción tiene:

- USAGE DISPLAY, cada identificador de la sentencia excepto el identificador POINTER debe tener USAGE DISPLAY, y cada literal de la sentencia debe ser alfanumérico
- USAGE NATIONAL, cada identificador de la sentencia excepto el identificador POINTER debe tener USAGE NATIONAL, y cada literal de la sentencia debe ser nacional
- USAGE DISPLAY-1, cada identificador de la sentencia excepto el identificador POINTER debe tener USAGE DISPLAY-1, y cada literal de la sentencia debe ser DBCS

Sólo se cambia la parte del campo de recepción en la que la sentencia STRING escribe los datos.

“Ejemplo: sentencia STRING” en la página 96

Tareas relacionadas
“Manejo de errores al unir y dividir series” en la página 177

Referencias relacionadas
sentencia STRING (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: sentencia STRING

El ejemplo siguiente muestra la sentencia STRING seleccionando y formateando información de un registro en una línea de salida.

El FILE SECTION define el registro siguiente:

```
01  RCD-01.
    05  CUST-INFO.
        10  CUST-NAME    PIC X(15).
        10  CUST-ADDR    PIC X(35).
    05  BILL-INFO.
        10  INV-NO       PIC X(6).
        10  INV-AMT      PIC $$,$$$.99.
        10  AMT-PAID     PIC $$,$$$.99.
        10  DATE-PAID    PIC X(8).
        10  BAL-DUE      PIC $$,$$$.99.
        10  DATE-DUE     PIC X(8).
```

El WORKING-STORAGE SECTION define los campos siguientes:

```
77  RPT-LINE            PIC X(120).
77  LINE-POS            PIC S9(3).
77  LINE-NO             PIC 9(5) VALUE 1.
77  DEC-POINT           PIC X VALUE ".".
```

El registro RCD-01 contiene la información siguiente (el símbolo *b* indica un espacio en blanco):

```
J.B.bSMITHbbbbb
444bSPRINGbST.,bCHICAGO,bILL.bbbbbb
A14275
$4,736.85
$2,400.00
09/22/76
$2,336.85
10/22/76
```

En PROCEDURE DIVISION, estos valores se producen antes de la sentencia STRING :

- RPT-LINE se establece en SPACES.
- LINE-POS, el elemento de datos que se utilizará como campo POINTER , se establece en 4.

A continuación se muestra la sentencia STRING :

```
STRING
   LINE-NO SPACE CUST-INFO INV-NO SPACE DATE-DUE SPACE
      DELIMITED BY SIZE
   BAL-DUE
      DELIMITED BY DEC-POINT
   INTO RPT-LINE
   WITH POINTER LINE-POS.
```

Puesto que el POINTER campo LINE-POS tiene el valor 4 antes de que se realice la sentencia STRING , los datos se mueven al campo receptor RPT-LINE empezando por la posición de carácter 4. Los caracteres de las posiciones 1 a 3 no se modifican.

Los elementos de envío que especifican DELIMITED BY SIZE se mueven en su totalidad al campo de recepción. Puesto que BAL-DUE está delimitado por DEC-POINT, el movimiento de BAL-DUE al campo de recepción se detiene cuando se encuentra una coma decimal (el valor de DEC-POINT).

Resultados de STRING

Cuando se realiza la sentencia STRING , los elementos se mueven a RPT-LINE tal como se muestra en la tabla siguiente.

Artículo	Posiciones
LINE-NO	4 - 8
Espacio	9
CUST-INFO	10 - 59
INV-NO	60 - 65
Espacio	66
DATE-DUE	67 - 74
Espacio	75
Parte de BAL-DUE que precede a la coma decimal	76 - 81

Después de realizar la sentencia STRING , el valor de LINE-POS es 82 y RPT-LINE tiene los valores que se muestran a continuación.

División de elementos de datos (UNSTRING)

Utilice la sentencia UNSTRING para dividir un campo de envío en varios campos de recepción. Una sentencia UNSTRING puede ocupar el lugar de varias sentencias MOVE .

Acerca de esta tarea

En la sentencia UNSTRING puede especificar:

- Delimitadores que, cuando se encuentra uno de ellos en el campo de envío, hacen que el campo de recepción actual deje de recibir y que el siguiente, si hay alguno, empiece a recibir (DELIMITED BY frase)
- Un campo para el delimitador que, cuando se encuentra en el campo de envío, hace que el campo de recepción actual deje de recibir (DELIMITER IN frase)
- Un elemento de datos entero que almacena el número de caracteres colocados en el campo de recepción actual (COUNT IN frase)
- Un elemento de datos entero que indica la posición de carácter más a la izquierda dentro del campo de envío en el que debe empezar el proceso de UNSTRING (fraseWITH POINTER)
- Un elemento de datos entero que almacena un recuento del número de campos de recepción sobre los que se actúa (fraseTALLYING IN)
- Acción que se debe realizar si se rellenan todos los campos de recepción antes de que se alcance el final del elemento de datos de envío (ON OVERFLOW frase)

El elemento de datos de envío y los delimitadores de la frase DELIMITED BY deben ser de categoría alfabética, alfanumérica, alfanumérica editada, DBCS, nacional o nacional editada.

Los elementos de datos de recepción pueden ser de categoría alfabética, alfanumérica, numérica, DBCS o nacional. Si es numérico, un elemento de datos de recepción debe ser decimal con zona o decimal nacional. Si un elemento de datos de recepción tiene:

- USAGE DISPLAY, el elemento emisor y cada elemento delimitador de la sentencia deben tener USAGE DISPLAY, y cada literal de la sentencia debe ser alfanumérico
- USAGE NATIONAL, el elemento emisor y cada elemento delimitador de la sentencia deben tener USAGE NATIONAL, y cada literal de la sentencia debe ser nacional
- USAGE DISPLAY-1, el elemento emisor y cada elemento delimitador de la sentencia deben tener USAGE DISPLAY-1, y cada literal de la sentencia debe ser DBCS

"Ejemplo: sentencia UNSTRING" en la página 98

Conceptos relacionados
"Unicode y la codificación de caracteres de idioma" en la página 190

Tareas relacionadas
"Manejo de errores al unir y dividir series" en la página 177

Referencias relacionadas
sentencia UNSTRING (*COBOL for Linux en x86 Consulta de lenguaje*)
Clases y categorías de datos (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: sentencia UNSTRING

El ejemplo siguiente muestra la sentencia UNSTRING transfiriendo información seleccionada de un registro de entrada. Parte de la información se organiza para su impresión y parte para su posterior procesamiento.

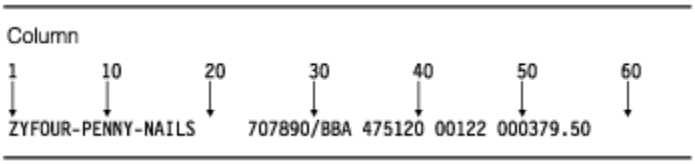
El FILE SECTION define los registros siguientes:

```
*  Record to be acted on by the UNSTRING statement:
 01  INV-RCD.
     05  CONTROL-CHARS              PIC XX.
     05  ITEM-INDENT                PIC X(20).
     05  FILLER                     PIC X.
     05  INV-CODE                   PIC X(10).
     05  FILLER                     PIC X.
     05  NO-UNITS                   PIC 9(6).
     05  FILLER                     PIC X.
     05  PRICE-PER-M                PIC 99999.
     05  FILLER                     PIC X.
     05  RTL-AMT                    PIC 9(6).99.
*
*  UNSTRING receiving field for printed output:
 01  DISPLAY-REC.
     05  INV-NO                     PIC X(6).
     05  FILLER                     PIC X VALUE SPACE.
     05  ITEM-NAME                  PIC X(20).
     05  FILLER                     PIC X VALUE SPACE.
     05  DISPLAY-DOLS               PIC 9(6).
*
*  UNSTRING receiving field for further processing:
 01  WORK-REC.
     05  M-UNITS                    PIC 9(6).
     05  FIELD-A                    PIC 9(6).
     05  WK-PRICE REDEFINES FIELD-A  PIC 9999V99.
     05  INV-CLASS                  PIC X(3).
*
*  UNSTRING statement control fields:
 77  DBY-1                          PIC X.
 77  CTR-1                          PIC S9(3).
 77  CTR-2                          PIC S9(3).
 77  CTR-3                          PIC S9(3).
 77  CTR-4                          PIC S9(3).
 77  DLTR-1                         PIC X.
 77  DLTR-2                         PIC X.
```

```
 77  CHAR-CT                      PIC S9(3).
 77  FLDS-FILLED                 PIC S9(3).
```

En PROCEDURE DIVISION, estos valores se producen antes de la sentencia UNSTRING :

- Un punto (.) se coloca en DBY-1 para su uso como delimitador.
- CHAR-CT (el campo POINTER) se establece en 3.
- El valor cero (0) se coloca en FLDS-FILLED (el campo TALLYING).
- Los datos se leen en el registro INV-RCD, cuyo formato es el que se muestra a continuación.



A continuación se muestra la sentencia UNSTRING :

```
* Move subfields of INV-RCD to the subfields of DISPLAY-REC
* and WORK-REC:
     UNSTRING INV-RCD
       DELIMITED BY ALL SPACES  OR "/"  OR DBY-1
       INTO ITEM-NAME    COUNT IN CTR-1
            INV-NO       DELIMITER IN DLTR-1  COUNT IN CTR-2
            INV-CLASS
            M-UNITS      COUNT IN CTR-3
            FIELD-A
            DISPLAY-DOLS DELIMITER IN DLTR-2  COUNT IN CTR-4
       WITH POINTER CHAR-CT
       TALLYING IN  FLDS-FILLED
       ON OVERFLOW  GO TO UNSTRING-COMPLETE.
```

Puesto que el POINTER campo CHAR-CT tiene el valor 3 antes de que se realice la sentencia UNSTRING , las dos posiciones de caracteres del campo CONTROL-CHARS en INV-RCD se ignoran.

Resultados de UNSTRING

Cuando se realiza la sentencia UNSTRING , se llevan a cabo los pasos siguientes:

1. Las posiciones 3 a 18 (FOUR-PENNY-NAILS) de INV-RCD se colocan en ITEM-NAME, se justifican a la izquierda en el área y las cuatro posiciones de caracteres no utilizadas se rellenan con espacios. El valor 16 se coloca en CTR-1.
2. Puesto que ALL SPACES está codificado como delimitador, los cinco caracteres de espacio contiguos en las posiciones 19 a 23 se consideran una aparición del delimitador.
3. Las posiciones de la 24 a la 29 (707890) se colocan en INV-NO. La barra inclinada del carácter delimitador (/) se coloca en DLTR-1y el valor 6 se coloca en CTR-2.
4. Las posiciones de la 31 a la 33 (BBA) se colocan en INV-CLASS. El delimitador es SPACE, pero debido a que no se ha definido ningún campo como área de recepción para delimitadores, se omite el espacio en la posición 34.
5. Las posiciones de la 35 a la 40 (475120) se colocan en M-UNITS. El valor 6 se coloca en CTR-3. El delimitador es SPACE, pero debido a que no se ha definido ningún campo como área de recepción para delimitadores, se omite el espacio en la posición 41.
6. Las posiciones de la 42 a la 46 (00122) se colocan en FIELD-A y se justifican a la derecha en el área. La posición de dígito de orden superior se rellena con un cero (0). El delimitador es SPACE, pero debido a que no se ha definido ningún campo como área de recepción para delimitadores, se omite el espacio en la posición 47.
7. Las posiciones de la 48 a la 53 (000379) se colocan en DISPLAY-DOLS. El delimitador de punto (.) en DBY-1 se coloca en DLTR-2, y el valor 6 se coloca en CTR-4.

8. Puesto que se ha actuado en todos los campos de recepción y no se han examinado dos caracteres en INV-RCD , se ejecuta la sentencia ON OVERFLOW . La ejecución de la sentencia UNSTRING se ha completado.

Después de realizar la sentencia UNSTRING , los campos contienen los valores que se muestran a continuación.

Campo	Valor
DISPLAY-REC	707890 FOUR-PENNY-NAILS 000379
WORK-REC	475120000122BBA
CHAR-CT (el campo POINTER)	55
FLDS-FILLED (el campo TALLYING)	6

Manipulación de series terminadas en nulo

Puede construir y manipular series terminadas en nulo (por ejemplo, series que se pasan a o desde un programa C) mediante diversos mecanismos.

Acerca de esta tarea

Por ejemplo, puede:

- Utilice constantes literales terminadas en nulo (Z". . . ").
- Utilice una sentencia INSPECT para contar el número de caracteres de una serie terminada en nulo:

```
MOVE 0 TO char-count
INSPECT source-field TALLYING char-count
                     FOR CHARACTERS
                     BEFORE X"00"
```

- Utilice una sentencia UNSTRING para mover caracteres de una serie terminada en nulo a un campo de destino y obtener el recuento de caracteres:

```
WORKING-STORAGE SECTION.
01  source-field          PIC X(1001).
01  char-count     COMP-5 PIC 9(4).
01  target-area.
    02 individual-char OCCURS 1 TO 1000 TIMES DEPENDING ON char-count
                         PIC X.
. . .
PROCEDURE DIVISION.
    UNSTRING source-field DELIMITED BY X"00"
                         INTO target-area
                         COUNT IN char-count
      ON OVERFLOW
        DISPLAY "source not null terminated or target too short"
    END-UNSTRING
```

- Utilice una sentencia SEARCH para localizar caracteres de espacio o nulos finales. Defina la serie que se está examinando como una tabla de caracteres únicos.
- Compruebe cada carácter de un campo de un bucle (PERFORM). Puede examinar cada carácter de un campo utilizando un modificador de referencia como source-field (I:1).

“Ejemplo: series terminadas en nulo” en la página 101

Tareas relacionadas

“Manejo de series terminadas en nulo” en la página 478

Referencias relacionadas

Literales alfanuméricos (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: series terminadas en nulo

El ejemplo siguiente muestra varias formas en las que puede procesar series terminadas en nulo.

```
 01  L pic X(20) value z'ab'.
 01  M pic X(20) value z'cd'.
 01  N pic X(20).
 01  N-Length pic 99 value zero.
 01  Y pic X(13) value 'Hello, World!'.
 . . .
* Display null-terminated string:
     Inspect N tallying N-length
       for characters before initial x'00'
     Display 'N: ' N(1:N-Length) ' Length: ' N-Length
     . . .
* Move null-terminated string to alphanumeric, strip null:
     Unstring N delimited by X'00' into X
     . . .
* Create null-terminated string:
     String Y     delimited by size
            X'00' delimited by size
            into N.
     . . .
* Concatenate two null-terminated strings to produce another:
     String L     delimited by x'00'
            M     delimited by x'00'
            X'00' delimited by size
            into N.
```

Referencia a subseries de elementos de datos

Consulte una subserie de un elemento de datos que tenga USAGE DISPLAY, DISPLAY-1o NATIONAL utilizando un modificador de referencia. También puede hacer referencia a una subserie de una serie de caracteres alfanuméricos o nacionales devuelta por una función intrínseca utilizando un modificador de referencia.

Acerca de esta tarea

El ejemplo siguiente muestra cómo utilizar un modificador de referencia para hacer referencia a una subserie de veinte caracteres de un elemento de datos denominado Customer-Record:

```
Move Customer-Record(1:20) to Orig-Customer-Name
```

Codifica un modificador de referencia entre paréntesis inmediatamente después del elemento de datos. Como muestra el ejemplo, un modificador de referencia puede contener dos valores separados por dos puntos, en este orden:

1. Posición ordinal (desde la izquierda) del carácter con el que desea que empiece la subserie
2. (Opcional) Longitud de la subserie necesaria en *posiciones de caracteres*

La posición y longitud del modificador de referencia para un elemento que tiene USAGE DISPLAY se expresan en términos de caracteres de un solo byte. La posición y longitud del modificador de referencia para elementos que tienen USAGE DISPLAY-1 o NATIONAL se expresan en términos de posiciones de caracteres DBCS y posiciones de caracteres nacionales, respectivamente.

Si omite la longitud en un modificador de referencia (codificando sólo la posición ordinal del primer carácter, seguida de dos puntos), la subserie se extiende hasta el final del elemento. Omita la longitud cuando sea posible como una técnica de codificación más simple y menos propensa a errores.

Puede hacer referencia a subseries de elementos de datos de USAGE DISPLAY , incluidos grupos alfanuméricos, elementos de datos editados alfanuméricos, elementos de datos editados numéricos, elementos de datos de coma flotante de visualización y elementos de datos decimales con zona, utilizando modificadores de referencia. Cuando se hace referencia a cualquiera de estos elementos de datos, el resultado es de categoría alfanumérica. Cuando se hace referencia a un elemento de datos alfabéticos, el resultado es de categoría alfabética.

Puede hacer referencia a subseries de elementos de datos de USAGE NATIONAL , incluidos grupos nacionales, elementos de datos editados a nivel nacional, elementos de datos editados a nivel numérico, elementos de datos de coma flotante nacional y elementos de datos decimales nacionales, utilizando modificadores de referencia. Cuando se hace referencia a cualquiera de estos elementos de datos, el resultado es de categoría nacional. Por ejemplo, supongamos que define un elemento de datos decimal nacional como se indica a continuación:

```
01  NATL-DEC-ITEM  Usage National  Pic 999  Value 123.
```

Puede utilizar NATL-DEC-ITEM en una expresión aritmética porque NATL-DEC-ITEM es de categoría numérica. Pero no puede utilizar NATL-DEC-ITEM(2:1) (el carácter nacional 2, que en notación hexadecimal es NX"3200") en una expresión aritmética, porque es de categoría nacional.

Puede hacer referencia a subseries de entradas de tabla, incluidas las entradas de longitud variable, utilizando modificadores de referencia. Para hacer referencia a una subserie de una entrada de tabla, codifique la expresión de subíndice antes del modificador de referencia. Por ejemplo, supongamos que PRODUCT-TABLE es una tabla de series de caracteres codificada correctamente. Para mover D al cuarto carácter de la segunda serie de la tabla, puede codificar esta sentencia:

```
MOVE 'D' to PRODUCT-TABLE (2), (4:1)
```

Puede codificar uno o ambos de los dos valores en un modificador de referencia como una variable o como una expresión aritmética.

“Ejemplo: expresiones aritméticas como modificadores de referencia” en la página 103

Puesto que los identificadores de función numérica se pueden utilizar en cualquier lugar donde se puedan utilizar expresiones aritméticas, puede codificar un identificador de función numérica en un modificador de referencia como la posición de carácter más a la izquierda o como la longitud, o ambos.

“Ejemplo: funciones intrínsecas como modificadores de referencia” en la página 104

Cada número del modificador de referencia debe tener un valor de al menos 1. La suma de los dos números no debe superar la longitud total del elemento de datos en más de 1 posición de carácter para que no haga referencia más allá del final de la subserie.

Si la posición de carácter más a la izquierda o el valor de longitud es un no entero de punto fijo, se produce un truncamiento para crear un entero. Si cualquiera de los dos es un no entero de coma flotante, se produce el redondeo para crear un entero.

Las opciones siguientes detectan modificadores de referencia fuera de rango y señalan infracciones con un mensaje de tiempo de ejecución:

- Opción de compilador SSRANGE
- Opción de tiempo de ejecución CHECK

Conceptos relacionados
“Modificadores de referencia” en la página 103
“Unicode y la codificación de caracteres de idioma” en la página 190

Tareas relacionadas
“Cómo hacer referencia a un elemento de una tabla” en la página 62

Referencias relacionadas
“SRANGE” en la página 299
Modificación de referencia (*COBOL for Linux en x86 Consulta de lenguaje*)
Definiciones de función (*COBOL for Linux en x86 Consulta de lenguaje*)

Modificadores de referencia

Los modificadores de referencia le permiten hacer referencia fácilmente a una subserie de un elemento de datos.

Por ejemplo, supongamos que desea recuperar la hora actual del sistema y visualizar su valor en un formato expandido. Puede recuperar la hora actual con la sentencia ACCEPT , que devuelve las horas, minutos, segundos y centésimas de segundo en este formato:

```
HHMMSSss
```

Sin embargo, es posible que prefiera ver la hora actual en este formato:

```
HH:MM:SS
```

Sin modificadores de referencia, tendría que definir elementos de datos para ambos formatos. También tendría que escribir código para convertir de un formato a otro.

Con los modificadores de referencia, no es necesario que proporcione nombres para los subcampos que describen los elementos TIME . La única definición de datos que necesita es para el tiempo tal como la devuelve el sistema. Por ejemplo:

```
01  REFMOD-TIME-ITEM    PIC X(8).
```

El código siguiente recupera y amplía el valor de tiempo:

```
     ACCEPT REFMOD-TIME-ITEM FROM TIME.
     DISPLAY "CURRENT TIME IS: "
* Retrieve the portion of the time value that corresponds to
*   the number of hours:
       REFMOD-TIME-ITEM (1:2)
       ":"
* Retrieve the portion of the time value that corresponds to
*   the number of minutes:
       REFMOD-TIME-ITEM (3:2)
       ":"
* Retrieve the portion of the time value that corresponds to
*   the number of seconds:
       REFMOD-TIME-ITEM (5:2)
```

“Ejemplo: expresiones aritméticas como modificadores de referencia” en la página 103
“Ejemplo: funciones intrínsecas como modificadores de referencia” en la página 104

Tareas relacionadas

“Asignación de entrada desde una pantalla o archivo (ACCEPT)” en la página 31
“Referencia a subseries de elementos de datos” en la página 101
“Utilización de datos nacionales (Unicode) en COBOL” en la página 191

Referencias relacionadas

Modificación de referencia (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: expresiones aritméticas como modificadores de referencia

Supongamos que un campo contiene algunos caracteres justificados por la derecha y desea mover esos caracteres a otro campo en el que se justifiquen por la izquierda. Puede hacerlo utilizando modificadores de referencia y una sentencia INSPECT .

Supongamos que un programa tiene los datos siguientes:

```
01  LEFTY     PIC X(30).
```

```
01  RIGHTY    PIC X(30)  JUSTIFIED RIGHT.
01  I         PIC 9(9)   USAGE BINARY.
```

El programa cuenta el número de espacios iniciales y, utilizando expresiones aritméticas en un modificador de referencia, mueve los caracteres justificados a la derecha a otro campo, justificados a la izquierda:

```
MOVE SPACES TO LEFTY
MOVE ZERO TO I
INSPECT RIGHTY
   TALLYING I FOR LEADING SPACE.
IF I IS LESS THAN LENGTH OF RIGHTY THEN
   MOVE RIGHTY ( I + 1 : LENGTH OF RIGHTY - I ) TO LEFTY
END-IF
```

La sentencia MOVE transfiere caracteres de RIGHTY, empezando por la posición calculada como I + 1 para una longitud que se calcula como LENGTH OF RIGHTY - I, al campo LEFTY.

Ejemplo: funciones intrínsecas como modificadores de referencia

Puede utilizar funciones intrínsecas en modificadores de referencia si no conoce la posición o longitud más a la izquierda de una subserie durante la compilación.

Por ejemplo, el fragmento de código siguiente hace que una subserie de Customer-Record se mueva al elemento de datos WS-name. La subserie se determina en tiempo de ejecución.

```
05  WS-name        Pic x(20).
05  Left-posn      Pic 99.
05  I              Pic 99.
. . .
Move Customer-Record(Function Min(Left-posn I):Function Length(WS-name)) to WS-name
```

Si desea utilizar una función no entera en una posición que requiere una función entera, puede utilizar la función INTEGER o INTEGER-PART para convertir el resultado en un entero. Por ejemplo:

```
Move Customer-Record(Function Integer(Function Sqrt(I)): ) to WS-name
```

Referencias relacionadas
INTEGER (*COBOL for Linux en x86 Consulta de lenguaje*)
INTEGER-PART (*COBOL for Linux en x86 Consulta de lenguaje*)

Recuento y sustitución de elementos de datos (INSPECT)

Utilice la INSPECT sentencia para inspeccionar caracteres o grupos de caracteres en un elemento de datos y para sustituirlos opcionalmente.

Acerca de esta tarea

Utilice la sentencia INSPECT para realizar las tareas siguientes:

- Cuente el número de veces que aparece un carácter específico en un elemento de datos (fraseTALLYING).
- Rellene un elemento de datos o partes seleccionadas de un elemento de datos con caracteres especificados como espacios, asteriscos o ceros (fraseREPLACING).
- Convierta todas las apariciones de un carácter específico o serie de caracteres de un elemento de datos en caracteres de sustitución que especifique (fraseCONVERTING).

Puede especificar uno de los siguientes elementos de datos como el elemento que se va a inspeccionar:

- Un elemento elemental descrito explícita o implícitamente como USAGE DISPLAY, USAGE DISPLAY-1o USAGE NATIONAL

- Un elemento de grupo alfanumérico o un elemento de grupo nacional

Si el artículo inspeccionado tiene:

- USAGE DISPLAY, cada identificador de la sentencia (excepto el campo de recuento TALLYING) debe tener USAGE DISPLAY, y cada literal de la sentencia debe ser alfanumérico
- USAGE NATIONAL, cada identificador de la sentencia (excepto el campo de recuento TALLYING) debe tener USAGE NATIONAL, y cada literal de la sentencia debe ser nacional
- USAGE DISPLAY-1, cada identificador de la sentencia (excepto el campo de recuento TALLYING) debe tener USAGE DISPLAY-1, y cada literal de la sentencia debe ser un literal DBCS

"Ejemplos: sentencia INSPECT" en la página 105

Conceptos relacionados
"Unicode y la codificación de caracteres de idioma" en la página 190

Referencias relacionadas
sentencia INSPECT (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplos: sentencia INSPECT

Los ejemplos siguientes muestran algunos usos de la sentencia INSPECT para examinar y sustituir caracteres.

En el ejemplo siguiente, la sentencia INSPECT examina y sustituye caracteres en el elemento de datos DATA-2. El número de veces que se produce un cero inicial (0) en el elemento de datos se acumula en COUNTR. La primera instancia del carácter A que sigue a la primera instancia del carácter C se sustituye por el carácter 2.

```
77  COUNTR           PIC 9   VALUE ZERO.
01  DATA-2           PIC X(11).
. . .
    INSPECT DATA-2
      TALLYING COUNTR FOR LEADING "0"
      REPLACING FIRST "A" BY "2" AFTER INITIAL "C"
```

DATA-2 antes	COUNTR después de	DATA-2 después de
00ACADEMY00	2	00AC2DEMY00
0000ALABAMA	4	0000ALABAMA
CHATHAM0000	0	CH2THAM0000

En el ejemplo siguiente, la sentencia INSPECT examina y sustituye caracteres en el elemento de datos DATA-3. Cada carácter que precede a la primera instancia de una comilla (") se sustituye por el carácter 0.

```
77  COUNTR           PIC 9   VALUE ZERO.
01  DATA-3           PIC X(8).
. . .
    INSPECT DATA-3
      REPLACING CHARACTERS BY ZEROS BEFORE INITIAL QUOTE
```

DATA-3 antes	COUNTR después de	DATA-3 después de
456"ABEL	0	000"ABEL
ANDES"12	0	00000"12
"TWAS BR	0	"TWAS BR

El ejemplo siguiente muestra el uso de frases INSPECT CONVERTING con AFTER y BEFORE para examinar y sustituir caracteres en el elemento de datos DATA-4. Todos los caracteres que siguen a la primera instancia del carácter / pero que preceden a la primera instancia del carácter ? (si hay alguno) se convierten de minúsculas a mayúsculas.

```
01  DATA-4           PIC X(11).
. . .
    INSPECT DATA-4
      CONVERTING
         "abcdefghijklmnopqrstuvwxyz" TO
         "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
      AFTER INITIAL "/"
      BEFORE INITIAL"?"
```

DATA-4 antes	DATA-4 después de
a/five/?six	a/FIVE/?six
r/Rexx/RRRr	r/REXX/RRRR
zfour?inspe	zfour?inspe

Conversión de elementos de datos (funciones intrínsecas)

Puede utilizar funciones intrínsecas para convertir elementos de datos de serie de caracteres a varios otros formatos, por ejemplo, a mayúsculas o minúsculas, para invertir el orden, a números o a una página de códigos de otra.

Acerca de esta tarea

Puede utilizar las funciones intrínsecas NATIONAL-OF y DISPLAY-OF para convertir a y desde series nacionales (Unicode).

También puede utilizar la sentencia INSPECT para convertir caracteres.

“Ejemplos: sentencia INSPECT” en la página 105

Tareas relacionadas

“Cambio de mayúsculas/minúsculas (UPPER-CASE, LOWER-CASE)” en la página 106
“Transformación a orden inverso (REVERSE)” en la página 107
“Conversión a números (NUMVAL, NUMVAL-C)” en la página 107
“Conversión de una página de códigos a otra” en la página 108

Cambio de mayúsculas/minúsculas (UPPER-CASE, LOWER-CASE)

Puede utilizar las funciones intrínsecas UPPER-CASE y LOWER-CASE para cambiar fácilmente las mayúsculas y minúsculas de las series alfanuméricas, alfabéticas o nacionales.

Acerca de esta tarea

```
01  Item-1   Pic x(30)  Value "Hello World!".
01  Item-2   Pic x(30).
. . .
    Display Item-1
    Display Function Upper-case(Item-1)
    Display Function Lower-case(Item-1)
    Move Function Upper-case(Item-1) to Item-2
    Display Item-2
```

El código anterior muestra los mensajes siguientes en el dispositivo de salida lógica del sistema:

```
Hello World!
```

```
HELLO WORLD!
hello world!
HELLO WORLD!
```

Las sentencias DISPLAY no cambian el contenido real de `Item-1`, sino que sólo afectan al modo en que se visualizan las letras. Sin embargo, la sentencia MOVE hace que las letras mayúsculas sustituyan el contenido de `Item-2`.

La conversión utiliza la correlación de casos definida en el entorno local actual. La longitud del resultado de la función puede diferir de la longitud del argumento.

Tareas relacionadas
“Asignación de entrada desde una pantalla o archivo (ACCEPT)” en la página 31
“Visualización de valores en una pantalla o en un archivo (DISPLAY)” en la página 32

Transformación a orden inverso (REVERSE)

Puede invertir el orden de los caracteres de una serie utilizando la función intrínseca REVERSE .

Acerca de esta tarea

```
Move Function Reverse(Orig-cust-name) To Orig-cust-name
```

Por ejemplo, la sentencia anterior invierte el orden de los caracteres en `Orig-cust-name`. Si el valor inicial es JOHNSON*bbb*, el valor después de que se realice la sentencia es *bbb*NOSNHOJ, donde *b* representa un espacio en blanco.

Conceptos relacionados
“Unicode y la codificación de caracteres de idioma” en la página 190

Conversión a números (NUMVAL, NUMVAL-C)

Las funciones NUMVAL y `NUMVAL-C` convierten series de caracteres (literales alfanuméricos o nacionales, o elementos de datos nacionales de clase o alfanuméricos de clase) en números. Utilice estas funciones para convertir números de representación de caracteres de formato libre a formato numérico para que pueda procesarlos numéricamente.

Acerca de esta tarea

Utilice `NUMVAL-C` cuando el argumento incluya un símbolo de moneda o una coma, o ambos, tal como se muestra en el ejemplo anterior. También puede colocar un signo algebraico antes o después de la serie de caracteres, y el signo se procesará. Los argumentos no deben superar los 18 dígitos al compilar con la opción predeterminada `ARITH(COMPAT)` (*modalidad de compatibilidad*) ni los 31 dígitos al compilar con `ARITH(EXTEND)` (*modalidad ampliada*), sin incluir los símbolos de edición.

NUMVAL, `NUMVAL-C` y devuelven valores de coma flotante largos (64 bits) en modalidad de compatibilidad y devuelven valores de precisión ampliada (128 bits) de coma flotante en modalidad ampliada. Una referencia a cualquiera de estas funciones representa una referencia a un elemento de datos numérico.

Como máximo 15 dígitos decimales se pueden convertir con precisión a coma flotante de precisión larga (como se describe en la referencia relacionada a continuación sobre conversiones y precisión). Si el argumento en NUMVAL, `NUMVAL-C`o tiene más de 15 dígitos, se recomienda que especifique la opción de compilador `ARITH(EXTEND)` para que se devuelva un resultado de función de precisión ampliada que pueda representar con precisión el valor del argumento.

Cuando se utiliza NUMVAL, NUMVAL-Co , no es necesario definir datos numéricos de forma estática en un formato fijo ni datos de entrada de forma precisa. Por ejemplo, supongamos que define los números que se van a especificar como se indica a continuación:

```
01  X   Pic S999V99  leading sign is separate.
. . .
    Accept X from Console
```

El usuario de la aplicación debe especificar los números exactamente tal como se define en la cláusula PICTURE . Por ejemplo:

```
+001.23
-300.00
```

Sin embargo, utilizando la función NUMVAL , podría codificar:

```
01  A   Pic x(10).
01  B   Pic S999V99.
. . .
    Accept A from Console
    Compute B = Function Numval(A)
```

La entrada podría ser:

```
1.23
-300
```

Conceptos relacionados
“Formatos para datos numéricos” en la página 39
“Conversiones de formato de datos” en la página 46
“Unicode y la codificación de caracteres de idioma” en la página 190

Tareas relacionadas
“Conversión a o desde representación nacional (Unicode)” en la página 199

Referencias relacionadas
“Conversiones y precisión” en la página 46
“HARIT” en la página 270

Conversión de una página de códigos a otra

Puede anidar las funciones intrínsecas DISPLAY-OF y NATIONAL-OF para convertir fácilmente de cualquier página de códigos a cualquier otra página de códigos.

Acerca de esta tarea

Por ejemplo, el código siguiente convierte una serie EBCDIC en una serie ASCII:

```
  77  EBCDIC-CCSID PIC 9(4) BINARY VALUE 1140.
  77  ASCII-CCSID  PIC 9(4) BINARY VALUE 819.
  77  Input-EBCDIC PIC X(80).
  77  ASCII-Output PIC X(80).
  . . .
* Convert EBCDIC to ASCII
      Move Function Display-of
          (Function National-of (Input-EBCDIC EBCDIC-CCSID),
              ASCII-CCSID)
      to ASCII-output
```

Conceptos relacionados
“Unicode y la codificación de caracteres de idioma” en la página 190

Tareas relacionadas
“Conversión a o desde representación nacional (Unicode)” en la página 199

Evaluación de elementos de datos (funciones intrínsecas)

Puede utilizar funciones intrínsecas para determinar la posición ordinal de un carácter en la secuencia de clasificación, para buscar el elemento más grande o más pequeño de una serie, para encontrar la longitud del elemento de datos o para determinar cuándo se ha compilado un programa.

Acerca de esta tarea

Utilice estas funciones intrínsecas:

- CHAR y ORD para evaluar enteros y caracteres alfabéticos o alfanuméricos individuales con respecto al orden de clasificación utilizado en un programa
- MAX, MIN, ORD-MAXy ORD-MIN para buscar los elementos más grandes y más pequeños de una serie de elementos de datos, incluidos los elementos de datos de USAGE NATIONAL
- LENGTH para buscar la longitud de los elementos de datos, incluidos los elementos de datos de USAGE NATIONAL
- WHEN-COMPILED para buscar la fecha y la hora en que se compiló un programa

Conceptos relacionados
“Unicode y la codificación de caracteres de idioma” en la página 190

Tareas relacionadas
“Evaluación de caracteres únicos para orden de clasificación” en la página 109
“Búsqueda del elemento de datos más grande o más pequeño” en la página 109
“Búsqueda de la longitud de los elementos de datos” en la página 112
“Determinación de la fecha de compilación” en la página 112

Evaluación de caracteres únicos para orden de clasificación

Para averiguar la posición ordinal de un carácter alfabético o alfanumérico determinado en el orden de clasificación, utilice la función ORD con el carácter como argumento. ORD devuelve un entero que representa esa posición ordinal.

Acerca de esta tarea

Puede utilizar una subserie de un carácter de un elemento de datos como argumento para ORD:

```
IF Function Ord(Customer-record(1:1)) IS > 194 THEN . . .
```

Si conoce la posición ordinal en la secuencia de clasificación de un carácter y desea encontrar el carácter al que corresponde, utilice la función CHAR con la posición ordinal entera como argumento. CHAR devuelve el carácter required . Por ejemplo:

```
INITIALIZE Customer-Name REPLACING ALPHABETIC BY Function Char(65)
```

Referencias relacionadas
CHAR (*COBOL for Linux en x86 Consulta de lenguaje*)
ORD (*COBOL for Linux en x86 Consulta de lenguaje*)

Búsqueda del elemento de datos más grande o más pequeño

Para determinar cuál de dos o más elementos de datos alfanuméricos, alfabéticos o nacionales tiene el valor más grande, utilice la función intrínseca MAX o ORD-MAX . Para determinar qué elemento tiene el valor más pequeño, utilice MIN o ORD-MIN. Estas funciones se evalúan según el orden de clasificación.

Acerca de esta tarea

Para comparar elementos numéricos, incluidos los que tienen USAGE NATIONAL, puede utilizar MAX, ORD-MAX, MINo ORD-MIN. Con estas funciones intrínsecas, se comparan los valores algebraicos de los argumentos.

Las funciones MAX y MIN devuelven el contenido de uno de los argumentos que proporcione. Por ejemplo, supongamos que el programa tiene las siguientes definiciones de datos:

```
05  Arg1   Pic x(10)  Value "THOMASSON ".
05  Arg2   Pic x(10)  Value "THOMAS    ".
05  Arg3   Pic x(10)  Value "VALLEJO   ".
```

La sentencia siguiente asigna VALLEJO*bbb* a las primeras posiciones de 10 caracteres de Customer-record, donde *b* representa un espacio en blanco:

```
Move Function Max(Arg1 Arg2 Arg3) To Customer-record(1:10)
```

Si ha utilizado MIN en su lugar, se asignará THOMAS*bbbb* .

Las funciones ORD-MAX y ORD-MIN devuelven un entero que representa la posición ordinal (contando desde la izquierda) del argumento que tiene el valor mayor o menor en la lista de argumentos que proporcione. Si ha utilizado la función ORD-MAX en el ejemplo anterior, el compilador emitiría un mensaje de error porque la referencia a una función numérica no está en un lugar válido. Utilizando los mismos argumentos que en el ejemplo anterior, ORD-MAX se puede utilizar como se indica a continuación:

```
Compute x = Function Ord-max(Arg1 Arg2 Arg3)
```

La sentencia anterior asigna el entero 3 a x si se utilizan los mismos argumentos que en el ejemplo anterior. Si ha utilizado ORD-MIN en su lugar, se devolverá el entero 2. Los ejemplos anteriores podrían ser más realistas si Arg1, Arg2y Arg3 fueran elementos sucesivos de una matriz (tabla).

Si especifica un elemento nacional para cualquier argumento, debe especificar todos los argumentos como clase nacional.

Tareas relacionadas
“Realización aritmética” en la página 49
“Proceso de elementos de tabla utilizando funciones intrínsecas” en la página 80
“Devolución de resultados de variables con funciones alfanuméricas o nacionales” en la página 110

Referencias relacionadas
MAX (*COBOL for Linux en x86 Consulta de lenguaje*)
MIN (*COBOL for Linux en x86 Consulta de lenguaje*)
ORD-MAX (*COBOL for Linux en x86 Consulta de lenguaje*)
ORD-MIN (*COBOL for Linux en x86 Consulta de lenguaje*)

Devolución de resultados de variables con funciones alfanuméricas o nacionales

Los resultados de las funciones alfanuméricas o nacionales pueden ser de longitudes y valores variables en función de los argumentos de la función.

Acerca de esta tarea

En el ejemplo siguiente, la cantidad de datos movidos a R3 y los resultados de la sentencia COMPUTE dependen de los valores y tamaños de R1 y R2:

```
01  R1    Pic x(10) value "e".
01  R2    Pic x(05) value "f".
01  R3    Pic x(20) value spaces.
```

```
01  L     Pic 99.
. . .
    Move Function Max(R1 R2) to R3
    Compute L = Function Length(Function Max(R1 R2))
```

Este código tiene los resultados siguientes:

- R2 se evalúa para que sea mayor que R1.
- La serie 'f*bbbb*' se mueve a R3, donde *b* representa un espacio en blanco. (Las posiciones de caracteres no rellenadas en R3 se rellenan con espacios.)
- L se evalúa en el valor 5.

Si R1 contenía 'g' en lugar de 'e', el código tendría los resultados siguientes:

- R1 se evaluaría como mayor que R2.
- La serie 'g*bbbbbbbbb*' se movería a R3. (Las posiciones de caracteres sin rellenar en R3 se rellenarían con espacios.)
- El valor 10 se asignaría a L.

Si un programa utiliza datos nacionales para argumentos de función, las longitudes y valores de los resultados de la función podrían variar del mismo modo. Por ejemplo, el código siguiente es idéntico al fragmento anterior, pero utiliza datos nacionales en lugar de datos alfanuméricos.

```
01  R1    Pic n(10) national value "e".
01  R2    Pic n(05) national value "f".
01  R3    Pic n(20) national value spaces.
01  L     Pic 99    national.
. . .
    Move Function Max(R1 R2) to R3
    Compute L = Function Length(Function Max(R1 R2))
```

Este código tiene los siguientes resultados, que son similares al primer conjunto de resultados excepto que son para caracteres nacionales:

- R2 se evalúa para que sea mayor que R1.
- La serie NX"6600 2000 2000 2000 2000" (el equivalente en caracteres nacionales de 'f*bbbb*', donde *b* representa un espacio en blanco), que se muestra aquí en notación hexadecimal con espacios añadidos para la legibilidad, se mueve a R3. Las posiciones de caracteres no rellenadas en R3 se rellenan con espacios nacionales.
- L se evalúa en el valor 5, la longitud en posiciones de caracteres nacionales de R2.

Es posible que esté tratando con la salida de longitud variable de las funciones alfanuméricas o nacionales . Planifique el programa en consecuencia. Por ejemplo, es posible que tenga que pensar en utilizar archivos de longitud variable cuando los registros que está grabando pueden tener longitudes diferentes:

```
File Section.
FD  Output-File Recording Mode V.
01  Short-Customer-Record  Pic X(50).
01  Long-Customer-Record   Pic X(70).
Working-Storage Section.
01  R1    Pic x(50).
01  R2    Pic x(70).
. . .
    If R1 > R2
      Write Short-Customer-Record from R1
    Else
      Write Long-Customer-Record from R2
    End-if
```

Tareas relacionadas

“Búsqueda del elemento de datos más grande o más pequeño” en la página 109
“Realización aritmética” en la página 49

Referencias relacionadas
MAX (*COBOL for Linux en x86 Consulta de lenguaje*)

Búsqueda de la longitud de los elementos de datos

Puede utilizar la función LENGTH en muchos contextos (incluidas tablas y datos numéricos) para determinar la longitud de un elemento. Por ejemplo, puede utilizar la función LENGTH para determinar la longitud de un literal alfanumérico o nacional, o un elemento de datos de cualquier tipo excepto DBCS.

Acerca de esta tarea

Función intrínseca LENGTH

La función LENGTH devuelve la longitud de un elemento nacional (un literal o cualquier elemento que tenga USAGE NATIONAL, incluidos los elementos de grupo nacional) como un entero igual a la longitud del argumento en posiciones de caracteres nacionales. Devuelve la longitud de cualquier otro elemento de datos como un entero igual a la longitud del argumento en posiciones de caracteres alfanuméricos.

La siguiente sentencia COBOL muestra cómo mover un elemento de datos al campo de un registro que contiene nombres de cliente:

```
Move Customer-name To Customer-record(1:Function Length(Customer-name))
```

LENGTH OF registro especial

También puede utilizar el registro especial LENGTH OF , que devuelve la longitud en bytes incluso para los datos nacionales. La codificación `Function Length(Customer-name)` o `LENGTH OF Customer-name` devuelve el mismo resultado para los elementos alfanuméricos: la longitud de `Customer-name` en bytes.

Puede utilizar las funciones LENGTH sólo cuando se permiten expresiones aritméticas. Sin embargo, puede utilizar el registro especial LENGTH OF en una mayor variedad de contextos. Por ejemplo, puede utilizar el registro especial LENGTH OF como argumento para una función intrínseca que acepte argumentos enteros. (No puede utilizar una función intrínseca como operando para el registro especial LENGTH OF .) También puede utilizar el registro especial LENGTH OF como parámetro en una sentencia CALL .

Tareas relacionadas
“Realización aritmética” en la página 49
“Creación de tablas de longitud variable (DEPENDING ON)” en la página 70
“Proceso de elementos de tabla utilizando funciones intrínsecas” en la página 80

Referencias relacionadas
“ADDR” en la página 268

LENGTH (*COBOL for Linux en x86 Consulta de lenguaje*)
LENGTH OF (*COBOL for Linux en x86 Consulta de lenguaje*)

Determinación de la fecha de compilación

Puede utilizar la función intrínseca WHEN-COMPILED para determinar cuándo se ha compilado un programa. El resultado de 21 caracteres indica el año, mes, día y hora de cuatro dígitos (en horas, minutos, segundos y centésimas de segundo) de la compilación, y la diferencia en horas y minutos de la hora media de Greenwich.

Acerca de esta tarea

Las primeras 16 posiciones están en el siguiente formato:

```
YYYYMMDDhhmmsshh
```

En su lugar, puede utilizar el registro especial `WHEN-COMPILED` para determinar la fecha y hora de la compilación en el formato siguiente:

```
MM/DD/YYhh.mm.ss
```

El registro especial `WHEN-COMPILED` sólo da soporte a un año de dos dígitos y no lleva fracciones de un segundo. Sólo puede utilizar este registro especial como campo de envío en una sentencia `MOVE` .

Referencias relacionadas
WHEN-COMPILADO (*COBOL for Linux en x86 Consulta de lenguaje*)

Capítulo 7. Procesando archivos

Acerca de esta tarea

La lectura de datos de archivos y la grabación de datos en archivos es una parte esencial de la mayoría de programas COBOL. El programa puede recuperar información, procesarla según lo solicite y, a continuación, escribir los resultados.

Sin embargo, antes del proceso, debe identificar los archivos y describir su estructura física, e indicar si están organizados como secuenciales, relativos, indexados o secuenciales de línea. La identificación de archivos implica nombrar los archivos y sus sistemas de archivos. También es posible que desee configurar un campo de estado de archivo que puede comprobar más adelante para verificar que el proceso ha funcionado correctamente.

Las tareas principales que puede realizar al procesar un archivo son primero abrir el archivo y, a continuación, leerlo, y (en función del tipo de organización y acceso de archivo) añadir, sustituir o suprimir registros.

Conceptos relacionados
“Terminología y conceptos de archivo” en la página 115
“Sistemas de archivos” en la página 122
“Grupos de datos de generación” en la página 130

Tareas relacionadas
“Identificación de archivos” en la página 117
“Especificación de una organización de archivos y modalidad de acceso” en la página 127
“Concatenación de archivos” en la página 139
“Apertura de archivos opcionales” en la página 140
“Configuración de un campo para el estado de archivo” en la página 140
“Descripción detallada de la estructura de un archivo” en la página 141
“Codificación de sentencias de entrada y salida para archivos” en la página 141
“Utilización de archivos de Db2” en la página 151
“Utilización de archivos QSAM” en la página 156
“Utilización de archivos SFS” en la página 156
“Utilización de archivos MongoDB” en la página 154

Terminología y conceptos de archivo

Los siguientes conceptos y terminología se utilizan en la información de COBOL para Linux sobre los archivos.

Nombre de archivo del sistema
El nombre de un archivo en un disco duro u otro soporte externo. Un nombre de archivo del sistema puede estar calificado por una vía de acceso u otro prefijo para garantizar la exclusividad. Existe un archivo en un sistema de archivos específico.

Los sistemas de archivos suelen proporcionar mandatos para gestionar archivos. El ejemplo siguiente muestra el uso del mandato `ls` para imprimir detalles sobre un archivo `Transaction.log` en el sistema de archivos STL y muestra la respuesta del sistema:

```
> ls -l Transaction.log
-rw-r--r--    1 cobdev   cobdev     6144 May 27 17:29 Transaction.log
```

Nombre de archivo interno
Palabra definida por el usuario que se especifica después de la palabra clave FD en una entrada de descripción de archivo en `FILE SECTION`y que se utiliza dentro de un programa para hacer referencia a un archivo.

En el ejemplo siguiente, LOG-FILE es un nombre de archivo interno:

```
Data division.
File section.
FD  LOG-FILE.
01  LOG-FILE-RECORD.
```

Los programas operan en archivos internos utilizando sentencias de E/S como OPEN, CLOSE, READ, WRITEy START. Como sugiere el término, un nombre de archivo interno no tiene ningún significado fuera de un programa.

La cláusula ASSIGN , que se describe a continuación, es el mecanismo que asocia un nombre de archivo interno con un nombre de archivo del sistema.

ID de sistema de archivos
Serie de tres caracteres que especifica el sistema de archivos en el que se almacena un archivo y a través del cual se accede a él.

Nombre de archivo externo
Nombre que actúa como intermediario entre un nombre de archivo interno y el nombre de archivo del sistema asociado. El nombre de archivo externo es visible fuera de un programa y normalmente se utiliza como el nombre de una variable de entorno que se establece en *file-information* (un ID de sistema de archivos opcional seguido del nombre de archivo del sistema) antes de que se ejecute el programa.

(Un nombre de archivo externo es distinto del nombre de un archivo externo, es decir, un archivo definido con la palabra clave EXTERNAL en su entrada FD .)

La cláusula ASSIGN asocia un nombre de archivo interno a un nombre de archivo del sistema y se especifica en el párrafo FILE-CONTROL . La cláusula ASSIGN tiene tres formatos básicos:

- ```
  SELECT internal-file-name ASSIGN TO user-defined-word
  ```
- ```
  SELECT internal-file-name ASSIGN TO 'literal'
  ```
- ```
  SELECT internal-file-name ASSIGN USING data-name
  . . .
      MOVE file-information TO data-name
  ```

palabra definida por el usuario y *literal* constan cada uno de un máximo de tres componentes, separados por guiones. De izquierda a derecha:

1. (Opcional) Comentario
2. (Opcional) ID de sistema de archivos
3. Nombre de archivo externo si se ha especificado una palabra definida por el usuario; nombre de archivo del sistema si se ha especificado un literal

file-information consta como máximo de dos componentes, separados por un guión. De izquierda a derecha:

1. (Opcional) ID de sistema de archivos
2. Nombre de archivo del sistema

Conceptos relacionados
“Sistemas de archivos” en la página 122

Tareas relacionadas
“Identificación de archivos” en la página 117

Referencias relacionadas
Cláusula ASSIGN (*COBOL for Linux en x86 Consulta de lenguaje*)
  ```

tipos de archivos

En un sistema Linux en x86 , hay dos tipos de archivos: archivos continuos y archivos binarios.

Archivos continuos
Los archivos continuos son archivos de texto. Los registros de los archivos continuos están separados por `/n`. Los archivos continuos se pueden ver mediante programas de utilidad del sistema operativo como, por ejemplo, `vi`, `cato lpr`. Los ejemplos de archivos continuos son conjuntos de datos secuenciales de registro (RSD) y secuenciales de línea (LSQ).

Si desea compartir datos con un proceso no COBOL, los datos deben estar en un archivo continuo.

Archivos binarios
Los archivos binarios son archivos en bruto. El contenido de los archivos binarios está totalmente controlado por las aplicaciones que utilizan el archivo. Los archivos binarios sólo pueden ser leídos por una aplicación que comprenda el formato. Ejemplos de archivos binarios son los archivos QSAM y VSAM.

Además, Db2, CICSy MongoDB proporcionan sistemas de archivos con los que el tiempo de ejecución COBOL puede comunicarse. Los archivos Db2, CICSy MongoDB son archivos binarios o puntos de montaje en bruto, que están totalmente gestionados por aplicaciones Db2, CICSy MongoDB . El contenido de los archivos Db2, CICSy MongoDB son totalmente invisibles para las herramientas proporcionadas por el sistema operativo; puede utilizar Db2, MongoDBo herramientas CICS como **`db2 show catalogue`**, **`mongodb show catalogue`**, **`sfsadmin list files`**o **`cics schema`** para ver el contenido.

Si está moviendo programas COBOL entre Linux en x86 y z/OS, es posible que también desee comprender los tipos de archivo de z/OS.

Hay dos tipos de archivo diferentes en z/OS:

Archivos secuenciales
Algunos ejemplos son los archivos QSAM, BSAM y EXCP.

Archivos relativos o indexados
Algunos ejemplos son los archivos BDAM y VSAM.

Conceptos relacionados
“Sistemas de archivos” en la página 122

Tareas relacionadas
“Identificación de archivos” en la página 117

Referencias relacionadas
Cláusula ASSIGN (*COBOL for Linux en x86 Consulta de lenguaje*)

Identificación de archivos

Para identificar un archivo, asocie el nombre de archivo que es interno del programa COBOL con el nombre de archivo del sistema correspondiente utilizando las cláusulas `SELECT` y `ASSIGN` en el párrafo `FILE-CONTROL` .

Acerca de esta tarea

Una forma simple de esta especificación es:

```
SELECT internalFilename ASSIGN TO fileSystemID-externalFilename
```

internalFilename especifica el nombre que utiliza dentro del programa para hacer referencia al archivo. El nombre interno normalmente no es el mismo que el nombre de archivo externo o el nombre de archivo del sistema.

En la cláusula `ASSIGN` , designa el nombre externo del archivo (*externalFilename*) al que desea acceder y, opcionalmente, especifica el sistema de archivos (*fileSystemID*) en el que existe o se va a crear el archivo.

Si codifica *fileSystemID* para identificar el sistema de archivos, utilice uno de los valores siguientes:

Db2
Db2 sistema de archivos de base de datos relacional.

LSQ
Sistema de archivos secuenciales de línea.

MONGO
Sistema de archivos de base de datos MongoDB .

QSAM
Sistema de archivos de método de acceso secuencial en cola.

RSD
Registre el sistema de archivos delimitado secuencial.

SFS
Sistema de archivos CICS Structured File Server.

STL
Sistema de archivos de idioma estándar.

VSA
Método de acceso de almacenamiento virtual, que implica el sistema de archivos SFS o STL .

SFS está implícito si la parte inicial (más a la izquierda) del nombre de archivo del sistema empieza por `/.:/cics/sfs`. De lo contrario, VSA implica el sistema de archivos STL .

Para los archivos `LINE SEQUENTIAL` , puede especificar o tomar como valor predeterminado `LSQ`, el sistema de archivos secuencial de línea.

Para los archivos `INDEXED`, `RELATIVE`y `SEQUENTIAL` , puede especificar `Db2`, `MONGO`, `SFS`, o `STL`. Para los `SEQUENTIAL` archivos, `RSD` o `QSAM` también es una opción válida.

Si no especifica el sistema de archivos para un archivo determinado, su sistema de archivos se determina de acuerdo con la prioridad descrita en la referencia relacionada sobre la prioridad.

Asocie un nombre de archivo interno a un nombre de archivo del sistema utilizando uno de estos elementos en la cláusula `ASSIGN` , tal como se describe a continuación:

- Una palabra definida por el usuario
- Un literal
- Un nombre de datos

Identificación de archivos utilizando una palabra definida por el usuario:

Para asociar un nombre de archivo interno a un nombre de archivo del sistema utilizando una palabra definida por el usuario, puede codificar una cláusula `SELECT` y una cláusula `ASSIGN` en el formato siguiente:

```
SELECT internalFilename ASSIGN TO userDefinedWord
```

La asociación del nombre de archivo interno a un nombre de archivo del sistema se completa en tiempo de ejecución. El ejemplo siguiente muestra cómo se asocia el nombre de archivo interno `LOG-FILE` con el archivo `Transaction.log` en el sistema de archivos STL utilizando la variable de entorno TRANLOG:

```
SELECT LOG-FILE ASSIGN TO TRANLOG
. . .
export TRANLOG=STL-Transaction.log
```

Si una variable de entorno TRANLOG no se ha establecido o se ha establecido en la serie nula cuando se ejecuta una sentencia `OPEN` para `LOG-FILE` , `LOG-FILE` se asocia con un archivo denominado `TRANLOG` en el sistema de archivos predeterminado.

Identificación de archivos utilizando un literal:

Para asociar un nombre de archivo interno a un nombre de archivo del sistema utilizando un literal, puede codificar una cláusula SELECT y una cláusula ASSIGN en el formato siguiente:

```
SELECT internalFilename ASSIGN TO 'fileSystemID-systemFilename'
```

En el literal, especifique el nombre de archivo del sistema y, opcionalmente, el sistema de archivos.

El ejemplo siguiente muestra cómo puede asociar un nombre de archivo interno myFile con el archivo extFile en el servidor *sfsServer* en el sistema de archivos SFS :

```
SELECT myFile ASSIGN TO 'SFS-/.:/cics/sfs/sfsServer/extFile'
```

Puesto que el literal especifica explícitamente el sistema de archivos y el nombre de archivo del sistema, la asociación de archivo puede resolverse en el momento de la compilación.

Para obtener más detalles sobre la codificación de la cláusula ASSIGN , consulte la referencia relacionada adecuada.

Identificación de archivos utilizando un nombre de datos:

Para asociar un nombre de archivo interno a un nombre de archivo del sistema utilizando un nombre de datos, codifique una cláusula SELECT y una cláusula ASSIGN con el formato siguiente:

```
SELECT internalFilename ASSIGN USING dataName
```

Mueva la información del sistema de archivos y del nombre de archivo a la variable *dataName* antes de que se procese el archivo.

El ejemplo siguiente muestra cómo puede asociar el nombre de archivo interno myFile con el archivo FebPayroll en el sistema de archivos STL :

```
SELECT myFile ASSIGN USING fileData
. . .
01 fileData PIC X(50).
. . .
    MOVE 'STL-FebPayroll' TO fileData
    OPEN INPUT myFile
```

La asociación se resuelve incondicionalmente en tiempo de ejecución.

Conceptos relacionados
“Sistemas de archivos” en la página 122
“Organización de archivo secuencial de línea” en la página 128
“Grupos de datos de generación” en la página 130

Tareas relacionadas
“Identificación de archivos en el sistema operativo (ASSIGN)” en la página 8
“Identificación de archivos de Db2” en la página 119
“Identificación de archivos SFS” en la página 120
“Identificación de archivos MongoDB” en la página 121
“Concatenación de archivos” en la página 139

Referencias relacionadas
“Prioridad de la determinación del sistema de archivos” en la página 121
“Variables de entorno de ejecución” en la página 234
Cláusula ASSIGN (*COBOL for Linux en x86 Consulta de lenguaje*)

Identificación de archivos de Db2

Para identificar un archivo en el sistema de archivos Db2 , especifique o de forma predeterminada el ID del sistema de archivos DB2.

Acerca de esta tarea

El nombre de archivo del sistema debe incluir el esquema para la tabla Db2 subyacente. Especifique el esquema directamente como prefijo del nombre de archivo del sistema.

Para la interoperación con TXSeries o CICS TX, utilice el nombre de esquema `CICS`. Por ejemplo, para asociar un archivo Db2 denominado TRANS en el esquema `CICS` con la variable de entorno EXTFILENAME, puede utilizar este mandato:

```
export EXTFILENAME=DB2-CICS.TRANS
```

De forma alternativa, para obtener más flexibilidad, especifique el sistema de archivos y el nombre de archivo del sistema por separado:

```
export COBRTOPT=FILESYS=DB2
export EXTFILENAME=CICS.TRANS
```

Para obtener más detalles sobre cómo utilizar los archivos DB2 , consulte la tarea relacionada adecuada a continuación.

Tareas relacionadas
“Identificación de archivos” en la página 117
“Utilización de archivos de Db2” en la página 151
“Utilización de códigos de estado del sistema de archivos” en la página 182
“Establecimiento de variables de entorno” en la página 229

Referencias relacionadas
“Sistema de archivos Db2” en la página 123
“ARCHIVOS” en la página 317

Identificación de archivos SFS

Para identificar un archivo en el sistema de archivos SFS, especifique o de forma predeterminada el ID del sistema de archivos SFS.

Acerca de esta tarea

El nombre de archivo del sistema debe empezar con el prefijo `SFS-` seguido por el nombre de servidor SFS y el nombre de archivo. Puede especificar el nombre de archivo del sistema si los archivos se encuentran en varios servidores SFS. El ejemplo siguiente muestra que el archivo del sistema denominado INVENTORY se encuentra en el servidor SFS denominado sfsServer.

```
export EXTFN=SFS-/.:/cics/sfs/sfsServer/INVENTORY
```

Si establece la variable de entorno CICS_TK_SFS_SERVER en el servidor SFS necesario , puede utilizar una especificación abreviada para el nombre de archivo del sistema en lugar de utilizar el nombre completo. El nombre de archivo del sistema tiene como prefijo el valor CICS_TK_SFS_SERVER, seguido de una barra inclinada, para crear el nombre de archivo completo del sistema. Por ejemplo:

```
export CICS_TK_SFS_SERVER=/.:/cics/sfs/sfsServer
export EXTFN=SFS-INVENTORY
```

El siguiente mandato `export` muestra un ejemplo más complejo de cómo puede establecer una variable de entorno MYFILE para identificar un archivo SFS indexado que tiene dos índices alternativos:

```
export MYFILE="/.:/cics/sfs/sfsServer/mySFSfil(\
/.:/cics/sfs/sfsServer/mySFSfil;myaltfil1,\
/.:/cics/sfs/sfsServer/mySFSfil;myaltfil2)"
```

El mandato proporciona la información siguiente:

- `/.:/cics/sfs/sfsServer` es el nombre completo del servidor CICS .
- *mySFSfil* es el archivo SFS base.
- `/.:/cics/sfs/sfsServer/mySFSfil` es el nombre de archivo completo del sistema base.
- *myaltfil1* y *myaltfil2* son los archivos de índice alternativos.

Para cada archivo de índices alternativo, el nombre de archivo debe estar en el formato de su nombre-archivo del sistema base completo seguido de un punto y coma (`;`) y el nombre del archivo de índices alternativo: `/.:/cics/sfs/sfsServer/mySFSfil;myaltfil1`.

Se necesita una coma entre las especificaciones de los archivos de índice alternativos en el mandato `export` .

Tareas relacionadas
“Identificación de archivos” en la página 117
“Utilización de archivos SFS” en la página 156
“Utilización de códigos de estado del sistema de archivos” en la página 182

Referencias relacionadas
“Sistema de archivos SFS” en la página 126

Identificación de archivos MongoDB

Para identificar un archivo en el sistema de archivos MongoDB , especifique o de forma predeterminada el ID del sistema de archivos `MONGO`.

Acerca de esta tarea

Para obtener más detalles sobre cómo utilizar los archivos MongoDB , consulte la tarea relacionada adecuada a continuación.

Tareas relacionadas
“Identificación de archivos” en la página 117
“Utilización de archivos MongoDB” en la página 154
“Utilización de códigos de estado del sistema de archivos” en la página 182
“Establecimiento de variables de entorno” en la página 229

Referencias relacionadas
“Sistema de archivos MongoDB” en la página 125
“ARCHIVOS” en la página 317

Prioridad de la determinación del sistema de archivos

El sistema de archivos aplicable a un archivo `SEQUENTIAL`, `INDEXED`o `RELATIVE` determinado se determina de acuerdo con la siguiente prioridad, de mayor a menor.

1. El sistema de archivos especificado por la variable de entorno de ejecución *nombre-asignación* o el valor del elemento de datos `USING` codificado en la cláusula `ASSIGN`
2. El sistema de archivos especificado por el componente situado más a la derecha del literal o palabra definida por el usuario que está codificado en la cláusula `ASSIGN` si dicho componente tiene al menos tres caracteres de longitud (y cumple los otros criterios descritos en la documentación de la cláusula `ASSIGN`)
3. El sistema de archivos predeterminado designado por la opción de tiempo de ejecución `FILESYS` (tal como se especifica en la variable de entorno de ejecución COBRTOPT)

Si no se determina ningún sistema de archivos por los medios anteriores, el sistema de archivos toma el valor predeterminado SFS si la parte más a la izquierda del nombre de archivo del sistema empieza por `/.:/cics/sfs`, de lo contrario, por STL.

Conceptos relacionados
“Sistemas de archivos” en la página 122

Tareas relacionadas
“Identificación de archivos en el sistema operativo (ASSIGN)” en la página 8
“Identificación de archivos” en la página 117

Referencias relacionadas
“Variables de entorno de ejecución” en la página 234
“ARCHIVOS” en la página 317
Cláusula ASSIGN (*COBOL for Linux en x86 Consulta de lenguaje*)

Sistemas de archivos

A los archivos orientados a registros que tienen una organización secuencial, relativa, indexada o secuencial de línea se accede a través de un *sistema de archivos*.

COBOL para Linux da soporte a los siguientes sistemas de archivos para archivos secuenciales, relativos e indexados:

Db2 (Db2 base de datos relacional) sistema de archivos
Permite a los programas COBOL crear y acceder a archivos almacenados en Db2.

Sistema de archivos MongoDB
Permite que los programas COBOL creen y accedan a documentos almacenados en MongoDB.

Sistema de archivos SFS (CICS Structured File Server)
Uno de los sistemas de archivos utilizados por CICS. CICS SFS se proporciona como parte de CICS. Los archivos SFS se pueden compartir con programas PL/I.

Sistema de archivos STL (lenguaje estándar)
Proporciona los recursos básicos para los archivos locales.

COBOL para Linux da soporte a los siguientes sistemas de archivos para archivos secuenciales:

Sistema de archivos QSAM (método de acceso secuencial en cola)
Permite a los programas COBOL acceder a archivos QSAM que se transfieren desde el sistema principal a Linux utilizando FTP.

sistema de archivos RSD (registro secuencial delimitado)
Permite que los programas COBOL compartan datos con programas escritos en otros lenguajes. Los archivos RSD sólo son secuenciales, con registros de longitud fija o variable, y dan soporte a todos los tipos de datos COBOL en los registros. La mayoría de los editores de archivos pueden editar los datos de texto de los registros.

Puede especificar el sistema de archivos para un archivo secuencial, relativo o indexado determinado de varias maneras. Para obtener detalles, consulte la referencia relacionada sobre la prioridad de la determinación del sistema de archivos.

Sólo se puede acceder a los archivos orientados a registros que tienen una organización secuencial de línea a través del sistema de archivos LSQ (secuencial de línea).

Los archivos Db2 se gestionan mediante el programa de utilidad de línea de mandatos DB2 ; los archivos SFS se gestionan mediante el programa de utilidad de línea de mandatos `sfsadmin` . Todos los demás archivos existen en el sistema de archivos secuencial de línea Linux , y están gestionados por mandatos estándar de Linux como `cp`, `ls`, `mvy` `rm`. (No obstante, no utilice el mandato `cp` o `mv` para los archivos SdU , que constan de varios archivos de componentes que se refieren entre sí internamente.)

Todos los sistemas de archivos le permiten utilizar sentencias COBOL para leer y escribir archivos COBOL. La mayoría de los programas tienen los mismos comportamientos en todos los sistemas de archivos. Sin embargo, los archivos grabados utilizando un sistema de archivos no se pueden leer utilizando un sistema de archivos diferente.

Para asociar un nombre de archivo COBOL a un sistema de archivos y un nombre de archivo de sistema operativo, utilice uno de los tres métodos siguientes que se listan en orden creciente de flexibilidad, donde RSD se utiliza como sistema de archivos de ejemplo:

- Código `ASSIGN TO RSD-`*`assignment-name`* en el programa.
- Establezca la siguiente variable de entorno: `export COBRTOPT=FS=RSD:`.
- Establezca el mandato siguiente: `export` *`assignment-name`*`=RSD-`*`os-file-name`*.

Es común que un programa COBOL utilice varios sistemas de archivos durante la ejecución; por ejemplo:

- Utilice el sistema de archivos RSD para los archivos de transacción que se utilizan para actualizar el archivo de base de datos.
- Utilice el sistema de archivos STL para archivos de base de datos indexados o relativos.
- Utilice el sistema de archivos RSD para los archivos de salida.
- Todos los demás archivos existen en el sistema de archivos nativo Linux .

Consejo: Para evitar cambiar y volver a compilar programas COBOL cuando el sistema de archivos cambia, puede utilizar variables de entorno para asociar un sistema de archivos y un nombre de archivo con la cláusula ASSIGN en lugar de utilizar los valores predeterminados de COBRTOPT.

Conceptos relacionados
"Organización de archivo secuencial de línea" en la página 128

Tareas relacionadas
"Identificación de archivos en el sistema operativo (ASSIGN)" en la página 8
"Identificación de archivos" en la página 117

Referencias relacionadas
"Prioridad de la determinación del sistema de archivos" en la página 121
"Sistema de archivos Db2" en la página 123
"Sistema de archivos QSAM" en la página 124
"Sistema de archivos RSD" en la página 125
"Sistema de archivos MongoDB" en la página 125
"Sistema de archivos SFS" en la página 126
"Sistema de archivos STL" en la página 126

Apéndice A, "Consideraciones sobre el formato de datos de host de IBM Z", en la página 549

Sistema de archivos Db2

El sistema de archivos Db2 admite archivos secuenciales, indexados y relativos. Proporciona una interoperación mejorada con TXSeries o CICS TX, lo que permite a los programas COBOL por lotes acceder a los archivos CICS ESDS, KSDS y RRDS almacenados en Db2.

La implementación del sistema de archivos Db2 garantiza que cada operación COBOL se confirme en la base de datos para que no se muestre ninguna semántica transaccional o de otra base de datos a través del programa COBOL.

El sistema de gestión de bases de datos Db2 (DBMS) proporciona funciones de copia de seguridad, compresión, cifrado y programa de utilidad, y también proporciona a los usuarios de Db2 un protocolo de mantenimiento y administración familiar.

El programa de utilidad de línea de mandatos db2 proporciona funciones administrativas para los archivos de Db2 . Por ejemplo, puede utilizar el mandato db2 `describe` para imprimir detalles sobre un archivo denominado `CICS.Transaction.log` en el sistema de archivos Db2 :

```
> db2 describe table CICS.\"Transaction.log\"
```

```
                        Data type                       Column
Column name             schema    Data type name     Length   Scale Nulls
----------------------- --------- ------------------ -------- ----- ----
RBA                     SYSIBM    CHARACTER               8      0 No
F1                      SYSIBM    CHARACTER              41      0 No
F2                      SYSIBM    VARCHAR                29      0 No
```

Para obtener más información sobre las funciones proporcionadas por el programa de utilidad db2 , especifique el mandato db2.

El sistema de archivos Db2 no es jerárquico.

Restricciones:

- Un programa determinado puede utilizar archivos Db2 en una sola base de datos.
- El sistema de archivos Db2 no es seguro para su uso con varias hebras.

Interoperación con TXSeries o CICS TX:

Para la interoperación con TXSeries o CICS TX, existen requisitos adicionales para los archivos Db2 :

- El nombre de esquema para la tabla Db2 debe ser `CICS`. Especifique el nombre completo en uno de estos elementos:
  - El literal `ASSIGN TO` , por ejemplo, `ASSIGN TO 'CICS.MYFILE'`
  - El valor de la variable de entorno, por ejemplo, `export ENVAR=CICS.MYFILE`
  - El valor de nombre de datos de `ASSIGN USING` , por ejemplo, `MOVE 'CICS.MYFILE' TO fileData`

  El resto del nombre de archivo después del esquema debe estar en mayúsculas.
- Los archivos de longitud fija tienen las longitudes de registro máximas siguientes:
  - Indexado (KSDS): 4005 bytes
  - Relativo (RRDS): 4001 bytes
  - Secuencial (ESDS): 4001 bytes
- Los archivos que tienen longitudes de registro mayores que las longitudes de registro máximas para archivos de longitud fija deben definirse como longitud variable.
- La longitud máxima de registro es de 32.767 bytes.
- Si un programa COBOL crea un archivo Db2 , la opción de tiempo de ejecución `FILEMODE(SMALL)` debe estar en vigor.

Conceptos relacionados
“Organización de archivos y modalidad de acceso” en la página 127

Tareas relacionadas
“Identificación de archivos de Db2” en la página 119
“Utilización de archivos de Db2” en la página 151

Referencias relacionadas

Efecto de la sentencia CLOSE en los tipos de archivo (*COBOL for Linux en x86 Consulta de lenguaje*)

DB2 Conceptos de administración de bases de datos y referencia de configuración (límites de SQL)

Sistema de archivos QSAM

El sistema de archivos QSAM (método de acceso secuencial en cola) admite registros fijos, variables y expandido. Utilizando el sistema de archivos QSAM, puede acceder directamente a un archivo QSAM que ha transferido desde el sistema principal a Linux. Los archivos QSAM dan soporte a todos los tipos de datos COBOL en el registro.

Puede obtener un archivo QSAM del sistema principal utilizando FTP con las opciones `binary` y `quote site rdw`. Si el archivo contiene datos de tipo carácter EBCDIC , compilar el programa Linux COBOL con `-host` para leer o escribir los datos de caracteres como EBCDIC. Si el archivo QSAM ya existe, puede cargar el mismo archivo en el sistema principal. Si el archivo no existe, debe crearlo utilizando los atributos de archivo correctos.

El sistema de archivos QSAM da soporte completo a las siguientes opciones de RECFM que se especifican en el sistema principal COBOL:

- `RECFM=V[B][S]`
- `RECFM=F[B][S]`

, donde:

- B representa registros bloqueados. B no tiene ninguna significación en Linux.
- F representa registros de longitud fija. Un archivo de registro de longitud fija QSAM no tiene metadatos. El archivo sólo tiene datos.
- S representa registros expandido.
- U representa registros de longitud no definida. U no es aplicable para archivos COBOL.
- V representa registros de longitud variable. Un archivo de registro de longitud variable QSAM tiene un RDW (Record Descriptor Word) inicial como prefijo para cada registro del archivo.

Conceptos relacionados
"Organización de archivos y modalidad de acceso" en la página 127

Tareas relacionadas
"Utilización de archivos QSAM" en la página 156

Sistema de archivos RSD

El sistema de archivos RSD (registro secuencial delimitado) soporta archivos secuenciales que tienen registros de longitud fija o variable . Puede procesar archivos RSD utilizando las funciones de programa de utilidad de archivos del sistema estándar como examinar, editar, copiar, suprimir e imprimir.

Los archivos RSD proporcionan un buen rendimiento. Le ofrecen la posibilidad de portar archivos fácilmente entre los sistemas basados en Linux y Ventanasy de compartir archivos entre programas y aplicaciones escritos en distintos idiomas.

Los archivos RSD dan soporte a todos los tipos de datos COBOL en registros de longitud fija o variable . Cada registro que se escribe va seguido de un carácter de control de nueva línea.

Conceptos relacionados
"Organización de archivos y modalidad de acceso" en la página 127

Sistema de archivos MongoDB

El sistema de archivos MongoDB admite archivos secuenciales, indexados y relativos.

La implementación del sistema de archivos MongoDB garantiza que cada operación COBOL se confirme en la base de datos para que no se muestre ninguna semántica transaccional o de otra base de datos a través del programa COBOL.

El programa de utilidad de línea de mandatos `mongo` proporciona funciones administrativas para los archivos MongoDB . Consulte MongoDB.

Restricción: Los archivos MongoDB no se pueden utilizar en los grupos de datos de generación (GDG).

Conceptos relacionados
"Organización de archivos y modalidad de acceso" en la página 127

Tareas relacionadas
“Identificación de archivos MongoDB” en la página 121
“Utilización de archivos MongoDB” en la página 154

Referencias relacionadas
MongoDB

Sistema de archivos SFS

El sistema de archivos CICS SFS (Structured File Server) es un sistema de archivos orientado a registros que da soporte a tres tipos de organización de archivos: secuencial (secuencia-entrada), relativa e indexada (en clúster). El sistema de archivos SFS proporciona los recursos básicos que necesita para acceder a los archivos de forma secuencial, aleatoria o dinámica.

Puede procesar archivos SFS utilizando las operaciones de archivo estándar como, por ejemplo, leer, escribir, reescribir, y suprimir.

Cada archivo SFS tiene un índice primario interno , que define el orden físico de los registros en el archivo, y puede tener cualquier número de índices secundarios, que proporcionan secuencias alternativas en las que se puede acceder a los registros.

Todos los datos de los archivos SFS están gestionados por un servidor SFS. SFS proporciona una herramienta del sistema, `sfsadmin`, para realizar funciones administrativas como crear archivos e índices, determinar qué volúmenes están disponibles en el servidor SFS, etc., a través de una interfaz de línea de mandatos. Para obtener detalles, consulte la publicación CICS en la referencia relacionada.

El sistema de archivos SFS no es jerárquico. Es decir, al identificar archivos SFS, solo puede especificar nombres de archivo individuales, no nombres de directorio, después del nombre de servidor.

El acceso de COBOL a los archivos SFS no es transaccional: cada operación en un archivo SFS es *atómica*, es decir, se realiza en su totalidad o no se realiza en absoluto. En el caso de una anomalía del sistema SFS, es posible que el resultado de una operación de archivo completada por una aplicación COBOL no se refleje en el archivo SFS.

El sistema de archivos SFS se ajusta a 85 COBOL Estándar.

Con Sistema de archivos SFS, puede leer y escribir fácilmente archivos para compartirlos con programas PL/I.

Restricciones:

- El sistema de archivos SFS no es seguro para su uso con varias hebras.
- No puede procesar archivos SFS utilizando programas COBOL para Linux de 64 bits.

Conceptos relacionados
“Organización de archivos y modalidad de acceso” en la página 127

Tareas relacionadas
“Identificación de archivos” en la página 117
“Identificación de archivos SFS” en la página 120
“Utilización de archivos SFS” en la página 156
“Mejora del rendimiento de SFS” en la página 159

Referencias relacionadas
Documentación de TXSeries
Documentación de CICS TX

Sistema de archivos STL

El sistema de archivos STL (sistema de archivos de lenguaje estándar) soporta archivos secuenciales, indexados y relativos. Proporciona los recursos de archivos básicos para acceder a los archivos.

El sistema de archivos STL se ajusta a 85 COBOL Estándary proporciona un buen rendimiento y la capacidad de portar fácilmente entre los sistemas basados en Linux y Ventanas.

Conceptos relacionados
"Organización de archivos y modalidad de acceso" en la página 127

Especificación de una organización de archivos y modalidad de acceso

En el párrafo `FILE-CONTROL` , debe definir la estructura física de un archivo y su modalidad de acceso, tal como se muestra a continuación.

Acerca de esta tarea

```
FILE-CONTROL.
    SELECT file ASSIGN TO FileSystemID-Filename
    ORGANIZATION IS org ACCESS MODE IS access.
```

Para *org*, puede elegir `SEQUENTIAL` (el valor predeterminado), `LINE SEQUENTIAL`, `INDEXED`o `RELATIVE`.

Para *access*, puede elegir `SEQUENTIAL` (el valor predeterminado), `RANDOM`o `DYNAMIC`.

Se debe acceder secuencialmente a los archivos secuenciales y secuenciales. Para archivos indexados o relativos, son posibles las tres modalidades de acceso.

Organización de archivos y modalidad de acceso

Puede organizar los archivos como secuenciales, secuenciales de línea, indexados o relativos. La modalidad de acceso define cómo COBOL lee y escribe archivos, pero no cómo se organizan los archivos.

Debe decidir sobre la organización de archivos y las modalidades de acceso cuando diseñe el programa.

La tabla siguiente resume la organización de archivos y las modalidades de acceso para archivos COBOL.

Tabla 9. ***Organización de archivos y modalidad de acceso***

Organización de archivo	**Orden de los registros**	**¿Se pueden suprimir o sustituir registros?**	**Modalidad de acceso**
secuencial	Orden en el que se escribieron	Un registro no se puede suprimir, pero su espacio se puede reutilizar para un registro de la misma longitud.	Sólo secuencial
Secuencial de línea	Orden en el que se escribieron	No se puede suprimir un registro, pero su espacio se puede reutilizar para un registro de la misma longitud.	Sólo secuencial
Indexado	Orden de clasificación por campo de clave	Sí	Secuencial, aleatorio o dinámico
relativo	Orden de los números de registro relativos	Sí	Secuencial, aleatorio o dinámico

El sistema de gestión de archivos maneja las solicitudes de entrada y salida y la recuperación de registros de los dispositivos de entrada-salida.

Conceptos relacionados
"Organización de archivos secuenciales" en la página 128
"Organización de archivo secuencial de línea" en la página 128

“Organización de archivos indexados” en la página 128
“Organización de archivos relativa” en la página 129
“Acceso secuencial” en la página 129
“Acceso aleatorio” en la página 129
“Acceso dinámico” en la página 129

Tareas relacionadas
“Especificación de una organización de archivos y modalidad de acceso” en la página 127

Organización de archivos secuenciales

Un archivo secuencial contiene registros organizados por el orden en el que se han especificado. El orden de los registros es fijo.

Los registros en archivos secuenciales sólo se pueden leer o grabar secuencialmente.

Después de colocar un registro en un archivo secuencial, no puede acortar, alargar o suprimir el registro. Sin embargo, puede actualizar (`REWRITE`) un registro si la longitud no cambia. Se añaden nuevos registros al final del archivo.

Si el orden en el que mantiene los registros en un archivo no es importante, la organización secuencial es una buena opción si hay muchos registros o sólo unos pocos. La salida secuencial también es útil para imprimir informes.

Conceptos relacionados
“Acceso secuencial” en la página 129

Referencias relacionadas
“Sentencias COBOL válidas para archivos secuenciales” en la página 144

Organización de archivo secuencial de línea

Los archivos secuenciales de línea son como los archivos secuenciales, excepto que los registros sólo pueden contener caracteres como datos. Los archivos secuenciales de línea están soportados por los archivos continuos de bytes nativos del sistema operativo.

Los archivos secuenciales de línea que se crean utilizando sentencias `WRITE` que tienen la frase `ADVANCING` se pueden dirigir a una impresora o a un disco.

Conceptos relacionados
“Organización de archivos secuenciales” en la página 128

Tareas relacionadas
“Identificación de archivos” en la página 117

Referencias relacionadas
“Sentencias COBOL válidas para archivos secuenciales de línea” en la página 145

Organización de archivos indexados

Un archivo indexado contiene registros ordenados por una *clave de registro*. Una clave de registro identifica de forma exclusiva un registro y determina la secuencia en la que se accede a él con respecto a otros registros.

Cada registro contiene un campo que contiene la clave de registro. Una clave de registro para un registro puede ser, por ejemplo, un número de empleado o un número de factura.

Un archivo indexado también puede utilizar *índices alternativos*, es decir, claves de registro que le permiten acceder al archivo utilizando una organización lógica diferente de los registros. Por ejemplo, puede acceder a un archivo a través del departamento de empleados en lugar de a través del número de empleado.

Las posibles modalidades de transmisión de registros (acceso) para archivos indexados son secuenciales, aleatorias o dinámicas. Cuando los archivos indexados se leen o se escriben secuencialmente, la secuencia es la de los valores de clave.

consideración EBCDIC: Al igual que con cualquier cambio en la secuencia de clasificación, si el archivo indexado es un archivo EBCDIC local, las claves EBCDIC no se reconocerán como tales fuera del programa COBOL. Por ejemplo, un programa de ordenación externo, a menos que también tenga soporte para EBCDIC, no ordenará los registros en el orden que podría esperar.

Referencias relacionadas
“Sentencias COBOL válidas para archivos indexados y relativos” en la página 145

Organización de archivos relativa

Un archivo de registro relativo contiene registros ordenados por su *clave relativa*, un número de registro que representa la ubicación del registro relativa a donde empieza el archivo.

Por ejemplo, el primer registro de un archivo tiene un número de registro relativo de 1, el décimo registro tiene un número de registro relativo de 10, etc. Los registros pueden tener longitud fija o longitud variable.

Las modalidades de transmisión de registros para archivos relativos son secuenciales, aleatorias o dinámicas. Cuando los archivos relativos se leen o se graban secuencialmente, la secuencia es la del número de registro relativo.

Referencias relacionadas
“Sentencias COBOL válidas para archivos indexados y relativos” en la página 145

Acceso secuencial

Para el acceso secuencial, codifique `ACCESS IS SEQUENTIAL` en el párrafo `FILE-CONTROL` .

Para los archivos indexados, se accede a los registros en el orden del campo de clave seleccionado (ya sea primario o alternativo), empezando por la posición actual del indicador de posición de archivo.

Para los archivos relativos, se accede a los registros en el orden de los números de registro relativos.

Conceptos relacionados
“Acceso aleatorio” en la página 129
“Acceso dinámico” en la página 129

Referencias relacionadas
“Indicador de posición de archivo” en la página 144

Acceso aleatorio

Para el acceso aleatorio, codifique `ACCESS IS RANDOM` en el párrafo `FILE-CONTROL` .

Para los archivos indexados, se accede a los registros según el valor que coloque en un campo de clave (primario, alternativo o relativo). Puede haber uno o más índices alternativos.

Para los archivos relativos, se accede a los registros según el valor que coloque en la clave relativa.

Conceptos relacionados
“Acceso secuencial” en la página 129
“Acceso dinámico” en la página 129

Acceso dinámico

Para el acceso dinámico, codifique `ACCESS IS DYNAMIC` en el párrafo `FILE-CONTROL` .

El acceso dinámico da soporte a una combinación de acceso secuencial y aleatorio en el mismo programa. Con el acceso dinámico, puede utilizar una definición de archivo COBOL para realizar un

proceso secuencial y aleatorio, accediendo a algunos registros en orden secuencial y a otros por sus claves.

Por ejemplo, supongamos que tiene un archivo indexado de registros de empleado, y el salario por hora del empleado forma la clave de registro. Además, suponga que su programa está interesado en aquellos empleados que ganan entre $12.00 y $18.00 por hora y aquellos que ganan $25.00 por hora y más. Para acceder a esta información, recupere el primer registro aleatoriamente (con una READde recuperación aleatoria) basándose en la clave de 1200. A continuación, empiece a leer secuencialmente (utilizando READ NEXT) hasta que el campo de salario supere los 1800. A continuación, vuelva a cambiar a una lectura aleatoria, esta vez basada en una clave de 2500. Después de esta lectura aleatoria, vuelva a leer secuencialmente hasta que llegue al final del archivo.

Conceptos relacionados
"Acceso secuencial" en la página 129
"Acceso aleatorio" en la página 129

Grupos de datos de generación

Un *grupo de datos de generación (GDG)* es una colección cronológica de archivos relacionados. Los GDG simplifican el procesamiento de múltiples versiones de datos relacionados.

Cada archivo dentro de un GDG se denomina un *conjunto de datos de generación (GDS)* o *generación*. (En esta información, los conjuntos de datos de generación se denominan *archivos de generación*. El término *archivo* en la estación de trabajo es equivalente al término *conjunto de datos* en el host.)

Dentro de un GDG, las generaciones pueden tener atributos similares o diferentes incluyendo ORGANIZATION, formato de registro y longitud de registro. Si todas las generaciones de un grupo tienen atributos coherentes y una organización secuencial, puede recuperar las generaciones juntas como un único archivo.

Existen ventajas al agrupar archivos relacionados. Por ejemplo:

- Se puede hacer referencia a los archivos del grupo mediante un nombre común.
- Los archivos del grupo se mantienen en orden de generación.
- Los archivos obsoletos se pueden descartar automáticamente.

Las generaciones dentro de un GDG han ordenado secuencialmente nombres relativos y absolutos que representan su edad.

El nombre relativo de un archivo de generación es el nombre de grupo seguido de un entero entre paréntesis. Por ejemplo, si el nombre de un grupo es hlq.PAY:

- hlq.PAY(0) hace referencia a la generación más actual.
- hlq.PAY(-1) hace referencia a la generación anterior.
- hlq.PAY(+1) especifica una nueva generación que se va a añadir.

El nombre absoluto de un archivo de generación contiene el número de generación y el número de versión. Por ejemplo, si el nombre de un grupo es hlq.PAY:

- hlq.PAY.g0005v00 hace referencia al archivo de generación 5, versión 0.
- hlq.PAY.g0006v00 hace referencia al archivo de generación 6, versión 0.

Para obtener más información sobre cómo formar nombres absolutos y relativos, consulte las tareas relacionadas.

El orden de generación suele ser, pero no necesariamente, el mismo que el orden en el que se han añadido los archivos a un grupo. En función de cómo añada archivos de generación utilizando nombres absolutos y relativos, puede insertar una generación en una posición inesperada en un grupo. Para obtener detalles, consulte la referencia relacionada sobre la inserción y el ajuste de los archivos de generación.

Los GDG están soportados en todos los sistemas de archivos COBOL para Linux excepto MONGO.

Restricción: Un GDG no puede contener un archivo indexado SFS que tenga índices alternativos o un archivo indexado SdU que requiera una lista de índices alternativos en el nombre de archivo. La restricción se debe a la ambigüedad entre la sintaxis de una lista de índices alternativos entre paréntesis y la sintaxis de nombres relativos GDG, que también requieren una expresión entre paréntesis.

Para obtener información sobre la creación e inicialización de grupos de datos de generación, consulte la tarea relacionada adecuada.

Para suprimir, reconstruir, limpiar, modificar o listar grupos de generación, o añadir o suprimir generaciones dentro de un grupo, utilice el programa de utilidad `gdgmgr` . Para ver un resumen de las funciones de `gdgmgr` , emita el mandato siguiente: `gdgmgr -h`. Para obtener más detalles sobre el programa de utilidad `gdgmgr` , consulte su `man` página mediante el mandato siguiente: `gdgmgr -man`.

Caso de uso

Un GDG se puede utilizar para almacenar y combinar datos para producir una aplicación de creación de informes diaria, mensual, trimestral o anual. La lista siguiente define cada frecuencia:

- Un máximo de 7 días a la semana
- Un máximo de 31 días en un mes
- Un máximo de 3 meses en un trimestre
- Un máximo de 4 trimestres en un año

Usted sigue escribiendo informes diarios automáticamente, pero sólo mantiene los 7 informes más recientes. El programa COBOL abre un archivo denominado `daily.reports(+1)`; el tiempo de ejecución generará un nombre exclusivo para el nuevo archivo y realizará cualquier limpieza necesaria para asegurarse de que las generaciones más antiguas se han envejecido correctamente. Si está resumiendo un informe anual en su lugar, el programa COBOL abre un archivo denominado `quarterly.reports(*)` para indicar que desea todos los datos y en orden secuencial.

Ejemplo

Siga estos pasos para crear diferentes versiones de GDG:

1. Exporte el nombre del sistema de archivos.
2. Compile y enlace el programa como de costumbre. Para obtener más información, consulte Capítulo 12, “Compilación, enlace y ejecución de programas”, en la página 229.
3. Cree un `gdg_test` base de GDG y liste el contenido utilizando este mandato: `gdgmgr -e -s -L 2 -c gdg_test -l`.

   La salida es la siguiente:

   ```
   GDG: gdg_test
     Catalogue   = ./gdg_test.catalogue
     Limit       = 2
     Days        = 0
     NoEmpty
     Scratch
     Entries     = 0
   ```

4. Ejecute el programa.
5. Cree distintas versiones de GDG y liste el contenido utilizando este mandato: `gdgmgr -l gdg_test`.

El archivo de origen es el siguiente:

```
cbl compile,pgmname(mixed)
ID DIVISION.
PROGRAM-ID. 'ins_gdg'.

ENVIRONMENT DIVISION.
INPUT-OUTPUT Section.

FILE-CONTROL.
    SELECT GDS_File
```

```
       ASSIGN using gds_filename
       ORGANIZATION is sequential.

DATA DIVISION.

FILE SECTION.
FD GDS_File
    Record contains 80 characters
    RECORDING MODE is F.
01 GDS_File-record pic x(80).

Working-Storage Section.
01 record-in            pic x(80) value spaces.
01 record-out           pic x(80) value spaces.
01 gds_filename         pic x(64) value spaces.

Linkage Section.

Procedure Division.
    move 0 to return-code.

    display "     Start ..."

    move 'gdg_test.g0001v00' to gds_filename.
    move '   Initial GDS 0001    [gdg_test.g0001v00]'
         to record-out.
    open output GDS_File
    write GDS_File-record from record-out
    close GDS_File
    move 'gdg_test(+1)' to gds_filename.
    move '   Increment of +1 (1) [gdg_test.g0002v00]'
         to record-out.
    open output GDS_File
    write GDS_File-record from record-out
    close GDS_File

    move 'gdg_test(+1)' to gds_filename.
    move '   Increment of +1 (2) [gdg_test.g0003v00]'
         to record-out.
    open output GDS_File
    write GDS_File-record from record-out
    close GDS_File
    display "     End   ..."

    goback.

END PROGRAM 'ins_gdg'.
```

Tareas relacionadas

“Creación de grupos de datos de generación” en la página 132
“Utilización de grupos de datos de generación” en la página 134

Referencias relacionadas

“Formato de nombre de los archivos de generación” en la página 135
“Inserción y acomodación de archivos de generación” en la página 136
“Limitar el proceso de grupos de datos de generación” en la página 138
"Especificación de archivo" en la *Guía de migración*

Creación de grupos de datos de generación

Para crear un grupo de datos de generación (GDG), primero cree su catálogo utilizando el mandato `gdgmgr` con el distintivo `-c` . (Un *catálogo GDG* es un archivo binario en Linux sistema de archivos nativo; un nombre de catálogo GDG tiene el formato *gdgBaseName*`.catalogue`.)

Acerca de esta tarea

A continuación, puede llenar el GDG con archivos de generación normalmente ejecutando programas COBOL que crean los archivos.

En el sistema de archivos LSQ de Linux :

Para crear un catálogo GDG en el sistema de archivos LSQ, RSD, SdU o STL, utilice el mandato `gdgmgr` con el distintivo `-c` . Por ejemplo, para crear el catálogo `./myGroups/transactionGroup.catalogue`, puede emitir este mandato:

```
gdgmgr -c ./myGroups/transactionGroup
```

El catálogo se crea de forma predeterminada en el directorio de trabajo actual (./). Opcionalmente, puede preceder el nombre de catálogo en el mandato `gdgmgr` con un nombre de vía de acceso, tal como se muestra anteriormente. El catálogo y los archivos de generación deben crearse en el mismo directorio.

En el sistema de archivos SFS:

Para crear un catálogo GDG en el sistema de archivos SFS, utilice el mandato `gdgmgr` con el distintivo `-c` . Puede especificar el nombre de archivo SFS en una forma completa o abreviada. Por ejemplo, el mandato siguiente crea un catálogo GDG utilizando un nombre SFS completo:

```
gdgmgr -c /.:/cics/sfs/sfsServer/baseName
```

En su lugar, puede especificar una forma abreviada del nombre de archivo SFS estableciendo primero la variable de entorno CICS_TK_SFS_SERVER y, a continuación, emitiendo el mandato `gdgmgr` utilizando también el distintivo `-F` para especificar el sistema de archivos SFS. Por ejemplo:

```
export CICS_TK_SFS_SERVER=/.:/cics/sfs/sfsServer
gdgmgr -F SFS -c baseName
```

Para alterar temporalmente el directorio de inicio de GDG predeterminado (~/gdg) para grupos SFS, establezca la variable de entorno gdg_home. Por ejemplo, los mandatos siguientes crean un catálogo de GDG `~/groups/forSFS/sfs/sfsServer/myGroup.catalogue`:

```
export gdg_home=~/groups/forSFS
gdgmgr -c /.:/cics/sfs/sfsServer/myGroup
```

Todos los archivos de generación SFS de un grupo determinado deben estar en el mismo servidor SFS.

En el sistema de archivos Db2 :

Para crear un catálogo GDG en el sistema de archivos Db2 , realice los pasos siguientes:

1. Inicialice el entorno de Db2 ejecutando el perfil para la instancia de Db2 que desea utilizar.
2. Establezca la variable de entorno DB2DBDFT en la base de datos para el grupo.
3. Utilice el mandato `gdgmgr` con el distintivo `-F` para especificar el sistema de archivos Db2 y con el distintivo `-c` . Especifique el esquema directamente en el nombre de catálogo necesario.

Por ejemplo, los mandatos siguientes crean el catálogo `~/gdg/db2/db2inst1/database/cics.dbGroup.catalogue`:

```
. /home/db2inst1/sqllib/db2profile
export DB2DBDFT=database
gdgmgr -F DB2 -c cics.dbGroup
```

El directorio de inicio de un catálogo GDG para los archivos Db2 se toma de la variable de entorno $gdg_home o, de lo contrario, toma el valor predeterminado ~/gdg.

Todos los archivos de generación de un grupo de datos de generación deben estar en una base de datos bajo un esquema.

Conceptos relacionados

“Sistemas de archivos” en la página 122
“Grupos de datos de generación” en la página 130

Tareas relacionadas
“Identificación de archivos de Db2” en la página 119
“Identificación de archivos SFS” en la página 120
“Utilización de grupos de datos de generación” en la página 134

Referencias relacionadas
“Limitar el proceso de grupos de datos de generación” en la página 138

Utilización de grupos de datos de generación

Para utilizar grupos de datos de generación, debe comprender cómo hacer referencia a ellos y cómo formar nombres absolutos y relativos de los archivos de generación dentro de un grupo.

Acerca de esta tarea

Consulte todo un grupo utilizando el nombre de grupo seguido opcionalmente por un asterisco entre paréntesis, por ejemplo: `abc.SALES(*)`. Cualquier formulario indica todos los archivos del grupo concatenados en orden de generación, la generación más actual en primer lugar. Para hacer referencia a una generación específica y añadirla a un grupo, utilice el nombre de grupo seguido de una referencia absoluta o relativa.

Nombres absolutos:

Para nombrar una generación específica de un grupo utilizando una referencia absoluta, utilice el nombre de grupo seguido de un número de generación absoluto y un número de versión. Por ejemplo, si el nombre de un grupo es `abc.SALES`:

- `abc.SALES.g0001v00` hace referencia al archivo de generación 1, versión 0.
- `abc.SALES.g0002v00` hace referencia al archivo de generación 2, versión 0.

Nombres relativos:

Para nombrar una generación específica de un grupo utilizando una referencia relativa, utilice el nombre de grupo seguido de un entero entre paréntesis. Por ejemplo, si el nombre de un grupo es `abc.SALES`:

- `abc.SALES(0)` hace referencia a la generación más actual.
- `abc.SALES(-1)` hace referencia a la generación anterior.
- `abc.SALES(+1)` especifica una nueva generación que se va a añadir.

Para obtener más detalles sobre los nombres absolutos y relativos, consulte la referencia relacionada sobre el formato de nombre de los archivos de generación.

Si una referencia absoluta o relativa es válida sintácticamente, el tiempo de ejecución determina si una referencia de datos de generación prevista hace referencia a un grupo de datos de generación o archivo de generación comprobando la existencia de un catálogo de grupos de datos de generación con un nombre adecuado. Si no se encuentra el catálogo, la referencia se trata como un identificador de archivo normal en lugar de como el nombre de un grupo de datos de generación o archivo de generación.

Una referencia relativa no firmada o negativa se resuelve en el identificador de archivo absoluto equivalente; se consulta al catálogo para determinar el equivalente. Si no hay ningún nombre absoluto en el catálogo que se corresponda con la referencia relativa resuelta, la referencia se trata como un identificador de archivo normal.

Una referencia relativa positiva firmada normalmente representa una nueva generación que se va a añadir al grupo. El incremento se añade al número de generación de la generación actual (cero) para formar el número de una nueva generación. Una referencia relativa positiva con signo es un medio alternativo para especificar el nombre absoluto equivalente.

Derivación de generación:

Si la suma del incremento y el número de generación actual es mayor que 9999, el nuevo número de generación se forma mediante ajuste (resta de 9999). Para ver ejemplos, consulte la referencia relacionada sobre la inserción y acomodación de archivos de generación.

Si un grupo no incluye una generación que tiene el nuevo número y se cumple alguna de las condiciones siguientes, se añade una nueva generación al grupo en la posición ordinal adecuada:

- El archivo no es `OPTIONAL` y la modalidad abierta es `OUTPUT`.
- El archivo es `OPTIONAL` y la modalidad de apertura es `I-O` o `EXTEND`.

Si un grupo ya contiene una generación que tiene el nuevo número, la generación existente se reutiliza y se puede sobrescribir si la modalidad abierta no es `INPUT`.

Conceptos relacionados
"Grupos de datos de generación" en la página 130

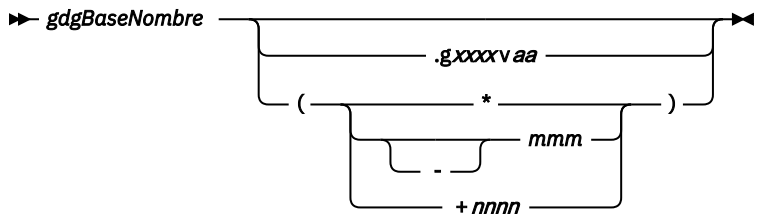
Tareas relacionadas
"Creación de grupos de datos de generación" en la página 132

Referencias relacionadas
"Formato de nombre de los archivos de generación" en la página 135
"Limitar el proceso de grupos de datos de generación" en la página 138
"Inserción y acomodación de archivos de generación" en la página 136
"Especificación de archivo" en la *Guía de migración*

Formato de nombre de los archivos de generación

La sintaxis siguiente describe cómo formar referencias absolutas y relativas a grupos de datos de generación (GDG) y archivos de generación.

Sintaxis de GDG y GDS



gdgBaseNombre
El nombre del grupo de datos de generación y el nombre base del catálogo GDG (un archivo binario *gdgBaseName*`.catalogue` en el sistema de archivos nativo de Linux).

gdgBaseNombre puede ser cualquier nombre de archivo válido. Sin embargo, evite especificar un nombre base que se parezca a una referencia absoluta o relativa, por ejemplo, `my(+1)Base`.

.gxxxxvaa
Generación absoluta y número de versión que identifican una generación específica, donde:

- *xxxx* es un número decimal de 4 dígitos sin signo de 0001 a 9999, ambos incluidos.
- *yy* es 00. 00 es el único número de versión soportado. Los números de versión distintos de cero se ignoran.

Por lo tanto, un nombre absoluto tiene el formato *gdgBaseName*`.g`*xxxx*`v`*yy*.

****
Sufijo relativo que designa un grupo entero. Todos los archivos de generación de un grupo se concatenan en orden de generación, la generación más actual en primer lugar.

Una referencia a todo el grupo tiene el formato *gdgBaseName*`(*)` o *gdgBaseName*.

mmm
Número de generación relativo de 0 a 999, inclusive, que identifica una generación específica. El número hace referencia a la generación actual (0), la generación anterior o menos actual (1) o (-1), etc. El signo negativo es opcional.

Es un error hacer referencia a una generación inexistente.

Por lo tanto, un nombre relativo para una generación existente tiene el formato *gdgBaseName*(*mmm*) o *gdgBaseName*(-*mmm*).

+nn
Un número de generación relativo positivo de 1 a 9999, inclusive, que normalmente identifica una generación que se va a crear y añadir a un GDG.

Un número mayor que 1 puede hacer que se omitan los números de generación. Por ejemplo, si sólo existe una generación y se especifica *gdgBaseName*(+3) , se omiten dos generaciones.

No es un error abrir para entrada una referencia como *gdgBaseName*(+1234) si la referencia se resuelve en un archivo existente en el grupo.

Conceptos relacionados
“Grupos de datos de generación” en la página 130

Tareas relacionadas
“Utilización de grupos de datos de generación” en la página 134

Referencias relacionadas
“Inserción y acomodación de archivos de generación” en la página 136
"Especificación de archivo" en la *Guía de migración*

Inserción y acomodación de archivos de generación

Una nueva generación en un GDG normalmente se inserta después de la generación actual y, por lo tanto, se convierte en la nueva generación actual.

Si la suma del incremento y el número de generación actual es mayor que 9999, se produce *ajuste* (resta de 9999) y el número de generación nuevo será menor que el número de generación actual. En este caso, la nueva generación puede insertarse antes de la generación actual, convirtiéndose en una generación anterior (menos actual) del grupo.

Considere el siguiente grupo inicial:

```
0: base.g0001v00
```

El número anterior a los dos puntos, denominado *número de época*, se utiliza para hacer que la inserción se produzca de forma previsible y aplica límites en los números de generación.

Normalmente, una nueva generación se inserta en el grupo en la posición indicada por el número de nueva generación después de que se haya producido un ajuste, y los números de época dentro del GDG no se cambian.

En cada uno de los ejemplos siguientes, la generación menos actual se lista en primer lugar y la generación más actual se lista en último lugar. *Cursiva* indica una nueva generación que se ha añadido al grupo.

Considere el grupo inicial de ejemplo que se muestra anteriormente. Si se especifica base(+1) , el número de generación actual (0001) se incrementa en 1. Como resultado, el grupo contendrá la nueva generación 0002 tal como se muestra a continuación:

```
0: base.g0001v00
0: base.g0002v00
```

Si se especifica base(+4) , el número de generación actual (0002) se incrementa en 4. El grupo como resultado contendrá la nueva generación (0006) como se muestra a continuación:

```
0: base.g0001v00
```

```
0: base.g0002v00
0: base.g0006v00
```

Si se especifica `base(+9997)`, el número de generación actual (0006) se incrementa en 9997. El número de generación resultante (10003) es mayor que 9999 y, por lo tanto, se ajusta para convertirse en 0004. En el grupo resultante, esta nueva generación (0004) se insertará antes de la generación actual (0006) como se muestra a continuación:

```
0: base.g0001v00
0: base.g0002v00
0: base.g0004v00
0: base.g0006v00
```

Después de esta última inserción, no es un error abrir `base(+9997)` para entrada, porque la referencia a `base(+9997)` indica el archivo existente `base.g0004v00` y no provoca ningún cambio en la estructura del grupo.

Normalmente, el número de época de una nueva generación es el mismo que el número de época de la generación actual. Sin embargo, hay dos casos en los que los números de época se ajustan, y la posición de inserción de la nueva generación es menos obvia:

- Si el número de generación actual es mayor o igual que 9000 y el número de generación nuevo es menor que 1000, se producirá la acomodación. Pero el número de época de la nueva generación aumentará para que la nueva generación pueda ser insertada después de la generación actual a pesar del hecho de que la nueva generación tiene un número de generación más bajo.

  Por ejemplo, considere el siguiente grupo inicial:

  ```
  0: base.g1000v00
  0: base.g9000v00
  ```

  Si se especifica `base(+1499)`, el número de generación actual (9000) se incrementa en 1499. El número de generación resultante (10499) es mayor que 9999 y, por lo tanto, se ajusta para convertirse en 0500. En el grupo resultante a la nueva generación (0500) se le da un número de época más alto y se convierte en la nueva generación actual a pesar de tener el número de generación más bajo en el grupo como se muestra:

  ```
  0: base.g1000v00
  0: base.g9000v00
  1: base.g0500v00
  ```

- Si el número de generación actual es menor que 1000 y el número de generación nuevo es mayor o igual que 9000, el número de época de la nueva generación se reduce a menos que el número de época de la generación actual ya sea cero. En este último caso el número de la época de las generaciones existentes se aumenta de modo que la nueva generación se insertará antes que todas las generaciones a pesar de tener un número de generación más alto.

  Por ejemplo, considere el siguiente grupo inicial:

  ```
  0: base.g0001v00
  0: base.g0999v00
  ```

  Si se especifica `base(+8501)`, la generación actual (0999) se incrementa en 8501. El número de generación resultante (9500) es menor que 9999; por lo tanto, no se produce ningún ajuste. El grupo resultante contendrá la nueva generación (9500) pero con el número de época 0. El número de época de las generaciones existentes se aumenta a 1. Como resultado, la nueva generación se convierte en la generación menos actual del grupo, tal como se muestra a continuación:

  ```
  0: base.g9500v00
  1: base.g0001v00
  1: base.g0999v00
  ```

Conceptos relacionados
“Grupos de datos de generación” en la página 130

Tareas relacionadas
“Utilización de grupos de datos de generación” en la página 134

Referencias relacionadas
“Formato de nombre de los archivos de generación” en la página 135
"Especificación de archivo" en la *Guía de migración*

Limitar el proceso de grupos de datos de generación

El proceso de límite se realiza siempre que se añade una nueva generación a un grupo de datos de generación, normalmente como resultado de una sentencia `OPEN` . También puede limitar el proceso manualmente utilizando la opción `-C` (limpieza) del programa de utilidad `gdgmgr` .

Si la opción `empty` está en vigor para un grupo, el proceso de límite elimina todos los archivos caducados del grupo. Si la opción `empty` no está en vigor para un grupo, los archivos caducados se eliminan del grupo, después de cualquier adición, en orden de generación menos actual a más actual hasta que el grupo esté en su límite.

Un archivo de generación se considera caducado si la diferencia entre la fecha del sistema actual y la fecha de creación del archivo supera el número de días especificado por la propiedad `days` del grupo.

Los archivos de generación que se eliminan de un grupo también se suprimen del sistema de archivos si la opción `scratch` está en vigor para el grupo.

Para grupos de Db2 y archivos SFS, la opción `-D` *`days`* no está soportada y la propiedad `days` del grupo se fuerza a cero. Por lo tanto, la antigüedad de estos archivos no es un factor durante el proceso de límite.

“Ejemplo: limitar el proceso” en la página 138

Conceptos relacionados
“Grupos de datos de generación” en la página 130

Tareas relacionadas
“Creación de grupos de datos de generación” en la página 132
“Utilización de grupos de datos de generación” en la página 134

Referencias relacionadas
"Especificación de archivo" en la *Guía de migración*

Ejemplo: limitar el proceso

El ejemplo siguiente muestra cómo afecta el proceso de límite al contenido de un grupo de datos de generación.

Considere un grupo de datos de generación, `audit`, que tiene un límite de tres generaciones, una opción `days` de 5 y la opción `noempty` , y que se ha creado con los siguientes cuatro nuevos archivos de generación de miembros:

```
0: audit.g0001v00
0: audit.g0003v00
0: audit.g0005v00
0: audit.g0007v00
```

Debido a que ninguno de los archivos estaba más allá de su fecha de caducidad cuando se creó el grupo, se permite que el grupo esté por encima del límite de tres generaciones. Pero después de siete días, todas las generaciones existentes están caducadas. Por lo tanto, si se añade una nueva generación, se eliminan las dos generaciones caducadas menos actuales para cumplir con el límite después de la adición.

Normalmente, la adición se realiza ejecutando un programa, pero el ejemplo siguiente muestra otra forma de añadir una generación y muestra el contenido de grupo resultante:

```
gdgmgr -a "audit(+2)" -1
```

```
0: audit.g0005v00
0: audit.g0007v00
0: audit.g0009v00
```

Si el grupo original tenía la opción `empty` en su lugar, el contenido del grupo después de la adición sólo contiene un archivo de generación, como se indica a continuación:

```
0: audit.g0009v00
```

Concatenación de archivos

En COBOL para Linux, puede concatenar varios archivos separando los identificadores de archivo individuales con dos puntos (`:`).

Acerca de esta tarea

Por ejemplo, el siguiente mandato `export` establece la variable de entorno MYFILE en STL-/home/*myUserID*/file1 seguido de STL-/home/*myUserID*/file2:

```
export MYFILE='STL-/home/myUserID/file1:STL-/home/myUserID/file2'
```

El mandato `export` junto con una cláusula SELECT y ASSIGN que asocian la variable de entorno con un nombre de archivo interno COBOL hace que los dos archivos se traten como un único archivo en el programa COBOL:

```
SELECT concatfile ASSIGN TO MYFILE
```

La concatenación está soportada si el *nombre-asignación* para la concatenación es una variable de entorno (como se muestra anteriormente), literal o nombre-datos `USING` .

Un archivo interno COBOL asignado a una concatenación de identificadores de archivo debe cumplir los criterios siguientes:

- `ORGANIZATION` es `SEQUENTIAL` o `LINE SEQUENTIAL`.
- `ACCESS MODE` es `SEQUENTIAL`.
- Las sentencias `OPEN` para el archivo tienen la modalidad `INPUT`.

Puede concatenar archivos que estén en cualquiera de los sistemas de archivos. Puede especificar el ID del sistema de archivos para cualquiera o todos los identificadores de archivo en una concatenación determinada. Sin embargo, todos los identificadores de archivo de una concatenación deben especificar o ser predeterminados en tiempo de ejecución en el mismo sistema de archivos, y todos los archivos deben tener atributos coherentes.

GDG: puede concatenar grupos de datos de generación completos (GDG) o archivos de generación individuales de uno o varios grupos. Si especifica un GDG en una concatenación, el archivo de generación más actual se lee primero, el siguiente más actual, y así sucesivamente hasta que se alcance el archivo de generación menos actual del grupo. Es un error incluir un GDG recién definido, y por lo tanto vacío, en una concatenación.

La validación de los identificadores de archivo individuales en una concatenación se aplaza hasta que se abra el archivo COBOL correspondiente. En ese momento, se resuelve el primer identificador de la concatenación y se intenta la apertura.

Después de un OPENsatisfactorio, el primer archivo de la concatenación se puede leer hasta que se haya alcanzado su último registro. En la siguiente sentencia READ , el siguiente identificador de archivo de la concatenación se resuelve y se abre.

Si todos los archivos de una concatenación no están disponibles cuando se ejecuta una sentencia OPEN para un archivo COBOL opcional, OPEN se ejecuta correctamente y la clave de estado de archivo se establece en 05. La primera operación READ devuelve el final del archivo y existe la condición AT END .

Conceptos relacionados
"Grupos de datos de generación" en la página 130

Tareas relacionadas
"Identificación de archivos" en la página 117
"Utilización de la condición de fin de archivo (AT END)" en la página 180

Referencias relacionadas

Cláusula ASSIGN (*COBOL for Linux en x86 Consulta de lenguaje*)
Concatenación de archivos (*COBOL for Linux en x86 Consulta de lenguaje*)

Apertura de archivos opcionales

Si un programa intenta abrir y leer un archivo que no existe, normalmente se produce un error.

Acerca de esta tarea

Sin embargo, puede haber ocasiones en las que abrir un archivo no existente tenga sentido. En estos casos, codifique la palabra clave OPTIONAL en la cláusula SELECT :

```
SELECT OPTIONAL file ASSIGN TO filename
```

Si un archivo está disponible u opcional afecta al proceso de OPEN , a la creación de archivos y a la clave de estado de archivo resultante. Por ejemplo, si abre en modalidad EXTEND, I-Oo INPUT un archivo noOPTIONAL no existente, el resultado es un error OPEN y se devuelve el estado de archivo 35. Sin embargo, si el archivo es OPTIONAL, la misma sentencia OPEN devuelve el estado de archivo 05 y, para las modalidades de apertura EXTEND y I-O, crea el archivo.

Tareas relacionadas
"Manejo de errores en operaciones de entrada y salida" en la página 178
"Utilización de claves de estado de archivo" en la página 181

Referencias relacionadas
Clave de estado de archivo (*COBOL for Linux en x86 Consulta de lenguaje*)
Notas de sentencia OPEN (*COBOL for Linux en x86 Consulta de lenguaje*)

Configuración de un campo para el estado de archivo

Establezca una clave de estado de archivo utilizando la cláusula FILE STATUS en el párrafo FILE-CONTROL y las definiciones de datos en DATA DIVISION.

Acerca de esta tarea

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    . . .
    FILE STATUS IS file-status
WORKING-STORAGE SECTION.
01  file-status  PIC 99.
```

Especifique la clave de estado de archivo *file-status* como un elemento nacional de categoría alfanumérica o de categoría de dos caracteres, o como un elemento decimal con zona de dos dígitos (USAGE DISPLAY) (como se muestra anteriormente) o decimal nacional (USAGE NATIONAL).

Restricción: el elemento de datos al que se hace referencia en la cláusula FILE STATUS no se puede localizar de forma variable; por ejemplo, no puede seguir a una tabla de longitud variable.

Tareas relacionadas
"Utilización de claves de estado de archivo" en la página 181

Referencias relacionadas
cláusula FILE STATUS (*COBOL for Linux en x86 Consulta de lenguaje*)

Descripción detallada de la estructura de un archivo

En el FILE SECTION de DATA DIVISION, inicie una descripción de archivo utilizando la palabra clave FD y el mismo nombre de archivo que ha utilizado en la cláusula SELECT correspondiente en el párrafo FILE-CONTROL .

Acerca de esta tarea

```
DATA DIVISION.
FILE SECTION.
FD  filename
01  recordname
    nn . . . fieldlength & type
    nn . . . fieldlength & type
    . . .
```

En el ejemplo anterior, *nombre_archivo* también es el nombre de archivo que utiliza en las sentencias OPEN, READy CLOSE .

recordname es el nombre del registro utilizado en las sentencias WRITE y REWRITE . Puede especificar más de un registro para un archivo.

fieldlength es la longitud lógica de un campo y *type* especifica el formato de un campo. Si rompe la entrada de descripción de registro más allá del nivel 01 de esta forma, correlacione cada elemento con precisión con el campo correspondiente del registro.

Referencias relacionadas
Relaciones de datos (*COBOL for Linux en x86 Consulta de lenguaje*)
Level-numbers (*COBOL for Linux en x86 Consulta de lenguaje*)
Cláusula PICTURE (*COBOL for Linux en x86 Consulta de lenguaje*)
Cláusula USAGE (*COBOL for Linux en x86 Consulta de lenguaje*)

Codificación de sentencias de entrada y salida para archivos

Después de identificar y describir los archivos en ENVIRONMENT DIVISION y DATA DIVISION, puede procesar los registros de archivo en PROCEDURE DIVISION del programa.

Acerca de esta tarea

Codifique el programa COBOL de acuerdo con los tipos de archivos que utilice, ya sean secuenciales, secuenciales de línea, indexados o relativos. El formato general para codificar la entrada y salida (como se muestra en el ejemplo al que se hace referencia a continuación) implica abrir el archivo, leerlo, escribir información en él y, a continuación, cerrarlo.

"Ejemplo: codificación COBOL para archivos" en la página 142

Tareas relacionadas
"Identificación de archivos" en la página 117

“Especificación de una organización de archivos y modalidad de acceso” en la página 127
“Apertura de archivos opcionales” en la página 140
“Configuración de un campo para el estado de archivo” en la página 140
“Descripción detallada de la estructura de un archivo” en la página 141
“Apertura de un archivo” en la página 144
“Lectura de registros de un archivo” en la página 146
“Adición de registros a un archivo” en la página 148
“Sustitución de registros en un archivo” en la página 149
“Supresión de registros de un archivo” en la página 149

Referencias relacionadas

“Indicador de posición de archivo” en la página 144
“Sentencias utilizadas al grabar registros en un archivo” en la página 147
“Sentencias PROCEDURE DIVISION utilizadas para actualizar archivos” en la página 150

Ejemplo: codificación COBOL para archivos

El ejemplo siguiente muestra el formato general de la codificación de entrada/salida. Las explicaciones de la información proporcionada por el usuario (texto en minúsculas en el ejemplo) se muestran después del código.

```
IDENTIFICATION DIVISION.
. . .
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT filename ASSIGN TO assignment-name  (1) (2)
    ORGANIZATION IS org ACCESS MODE IS access  (3) (4)
    FILE STATUS IS file-status  (5)
    . . .
DATA DIVISION.
FILE SECTION.
FD  filename
01  recordname  (6)
    nn . . . fieldlength & type  (7) (8)
    nn . . . fieldlength & type
    . . .
WORKING-STORAGE SECTION.
01  file-status    PIC 99.
    . . .
PROCEDURE DIVISION.
    OPEN iomode filename   (9)
    . . .
    READ filename
    . . .
    WRITE recordname
    . . .
    CLOSE filename
  STOP RUN.
```

(1) *nombre_archivo*

Cualquier nombre COBOL válido. Debe utilizar el mismo nombre de archivo en la cláusula SELECT y la entrada FD , y en las sentencias OPEN, READ, START, DELETEy CLOSE . Este nombre no es necesariamente el nombre de archivo del sistema. Cada archivo requiere su propia cláusula SELECT , entrada FD y sentencias de entrada/salida. Para WRITE y REWRITE, especifique un registro definido para el archivo.

(2) *nombre-asignación*

Puede codificar ASSIGN TO *nombre-asignación* para especificar directamente el ID del sistema de archivos de destino y el nombre de archivo del sistema, o puede establecer el valor indirectamente utilizando una variable de entorno.

Si desea que el nombre de archivo del sistema se identifique a la hora de OPEN , especifique ASSIGN USING *nombre-datos*. Se utiliza el valor de *nombre-datos* en el momento de la ejecución de la sentencia OPEN para dicho archivo. Opcionalmente, puede preceder el nombre de archivo del sistema por el identificador del sistema de archivos, utilizando un guión como separador.

El ejemplo siguiente muestra cómo `inventory-file` se asocia dinámicamente con el archivo `/user/inventory/parts` mediante una sentencia `MOVE` :

```
SELECT inventory-file ASSIGN USING a-file . . .
. . .
FD inventory-file . . .
. . .
77 a-file PIC X(25) VALUE SPACES.
. . .
    MOVE "/user/inventory/parts" TO a-file
    OPEN INPUT inventory-file
```

El ejemplo siguiente muestra cómo `inventory-file` se asocia dinámicamente con el archivo indexado CICS SFS `parts`, y muestra cómo los archivos de índice alternativos `altpart1` y `altpart2` están asociados con el nombre completo (`/.:/cics/sfs` en este ejemplo) del servidor CICS .

```
SELECT inventory-file ASSIGN USING a-file . . .
    ORGANIZATION IS INDEXED
    ACCESS MODE IS DYNAMIC
    RECORD KEY IS FILESYSFILE-KEY
    ALTERNATE RECORD KEY IS ALTKEY1
    ALTERNATE RECORD KEY IS ALTKEY2. . . .
. . .
FILE SECTION.
FD inventory-file . . .
. . .
WORKING-STORAGE SECTION.
01 a-file  PIC X(80). . .
. . .
    MOVE "/.:/cics/sfs/parts(/.:/cics/sfs/parts;altpart1,/.:/
       cics/sfs/parts;altpart2)" TO a-file
    OPEN INPUT inventory-file
```

(3) ***organización***
Indica la organización: `LINE SEQUENTIAL`, `SEQUENTIAL`, `INDEXED`o `RELATIVE`. Si omite esta cláusula, el valor predeterminado es `ORGANIZATION SEQUENTIAL`.

(4) ***acceso***
Indica la modalidad de acceso, `SEQUENTIAL`, `RANDOM`o `DYNAMIC`. Si omite esta cláusula, el valor predeterminado es `ACCESS SEQUENTIAL`.

(5) ***estado-archivo***
La clave de estado del archivo COBOL. Puede especificar la clave de estado de archivo como un elemento de datos alfanumérico o nacional de dos caracteres, o como un elemento decimal con zona o decimal nacional de dos dígitos.

(6) ***nombre_registro***
El nombre del registro utilizado en las sentencias `WRITE` y `REWRITE` . Puede especificar más de un registro para un archivo.

(7) ***longitud_campo***
Longitud lógica del campo.

(8) ***tipo***
Debe coincidir con el formato de registro del archivo. Si rompe la entrada de descripción de registro más allá de la descripción level-01 , correlacione cada elemento de forma precisa con los campos del registro.

(9) ***iomode***
Especifica la modalidad de apertura. Si sólo está leyendo de un archivo, codifique `INPUT`. Si sólo está grabando en un archivo, codifique `OUTPUT` (para abrir un archivo nuevo o grabar sobre uno existente) o `EXTEND` (para añadir registros al final del archivo). Si está realizando ambas acciones, codifique `I-O`.

Restricción: Para archivos secuenciales de línea, `I-O` no es una modalidad válida de la sentencia `OPEN` .

Indicador de posición de archivo

El indicador de posición de archivo marca el siguiente registro al que se debe acceder para solicitudes COBOL secuenciales.

No establece explícitamente el indicador de posición de archivo en ningún lugar del programa. Se establece mediante sentencias OPEN, START, READ, READ NEXTy READ PREVIOUS satisfactorias. Las solicitudes READ, READ NEXTo READ PREVIOUS posteriores utilizan la ubicación del indicador de posición de archivo establecido y la actualizan.

El indicador de posición de archivo no se utiliza ni se ve afectado por las sentencias de salida WRITE, REWRITEo DELETE. El indicador de posición de archivo no tiene significado para el proceso aleatorio.

Apertura de un archivo

Antes de que el programa pueda utilizar una sentencia WRITE, START, READ, REWRITEo DELETE para procesar registros en un archivo, el programa debe abrir primero el archivo utilizando una sentencia OPEN .

Acerca de esta tarea

```
PROCEDURE DIVISION.
    . . .
    OPEN iomode filename
```

En el ejemplo anterior, *iomode* especifica la modalidad abierta. Si sólo está leyendo desde el archivo, codifique INPUT para la modalidad abierta. Si sólo está grabando en el archivo, codifique OUTPUT (para abrir un archivo nuevo o grabar sobre uno existente) o EXTEND (para añadir registros al final del archivo) para la modalidad de apertura.

Para abrir un archivo que ya contiene registros, utilice OPEN INPUT, OPEN I-O (no válido para archivos secuenciales de línea) o OPEN EXTEND.

Si codifica OPEN OUTPUT para un archivo SdU o SFS que contiene registros, el tiempo de ejecución COBOL suprime el archivo y, a continuación, crea el archivo con los atributos proporcionados por COBOL. Si no desea que se suprima un archivo SdU o SFS, abra el archivo codificando OPEN I-O en su lugar.

Si abre un archivo secuencial, secuencial de línea o relativo como EXTEND, los registros añadidos se colocan después del último registro existente en el archivo. Si abre un archivo indexado como EXTEND, cada registro que añada debe tener una clave de registro superior al registro más alto del archivo.

Conceptos relacionados
“Organización de archivos y modalidad de acceso” en la página 127

Tareas relacionadas
“Apertura de archivos opcionales” en la página 140

Referencias relacionadas
“Sentencias COBOL válidas para archivos secuenciales” en la página 144
“Sentencias COBOL válidas para archivos secuenciales de línea” en la página 145
“Sentencias COBOL válidas para archivos indexados y relativos” en la página 145

sentencia OPEN (*COBOL for Linux en x86 Consulta de lenguaje*)

Sentencias COBOL válidas para archivos secuenciales

La tabla siguiente muestra las posibles combinaciones de sentencias de entrada-salida para archivos secuenciales. ' X' indica que la sentencia se puede utilizar con el modo abierto que se muestra en la parte superior de la columna.

Tabla 10. ***Sentencias COBOL válidas para archivos secuenciales***					
Modalidad de acceso	**sentencia COBOL**	**ABRIR ENTRADA**	**ABRIR SALIDA**	**OPEN I-O**	**AMPLIACIÓN ABIERTA**
secuencial	OPEN	X	X	X	X
	WRITE		X		X
	START				
	READ	X		X	
	REWRITE			X	
	DELETE				
	CLOSE	X	X	X	X

Conceptos relacionados
“Organización de archivos secuenciales” en la página 128
“Acceso secuencial” en la página 129

Sentencias COBOL válidas para archivos secuenciales de línea

La tabla siguiente muestra las posibles combinaciones de sentencias de entrada-salida para archivos secuenciales de línea. ' X' indica que la sentencia se puede utilizar con el modo abierto que se muestra en la parte superior de la columna.

Tabla 11. ***Sentencias COBOL válidas para archivos secuenciales de línea***					
Modalidad de acceso	**sentencia COBOL**	**ABRIR ENTRADA**	**ABRIR SALIDA**	**OPEN I-O**	**AMPLIACIÓN ABIERTA**
secuencial	OPEN	X	X		X
	WRITE		X		X
	START				
	READ	X			
	REWRITE				
	DELETE				
	CLOSE	X	X		X

Conceptos relacionados
“Organización de archivo secuencial de línea” en la página 128
“Acceso secuencial” en la página 129

Sentencias COBOL válidas para archivos indexados y relativos

La tabla siguiente muestra las posibles combinaciones de sentencias de entrada-salida para archivos indexados y relativos. ' X' indica que la sentencia se puede utilizar con el modo abierto que se muestra en la parte superior de la columna.

Tabla 12. ***Sentencias COBOL válidas para archivos indexados y relativos***					
Modalidad de acceso	**sentencia COBOL**	**ABRIR ENTRADA**	**ABRIR SALIDA**	**OPEN I-O**	**AMPLIACIÓN ABIERTA**
secuencial	OPEN	X	X	X	X
	WRITE		X		X
	START	X		X	
	READ	X		X	
	REWRITE			X	
	DELETE			X	
	CLOSE	X	X	X	X
Aleatorio	OPEN	X	X	X	
	WRITE		X	X	
	START				
	READ	X		X	
	REWRITE			X	
	DELETE			X	
	CLOSE	X	X	X	
dinámico	OPEN	X	X	X	
	WRITE		X	X	
	START	X		X	
	READ	X		X	
	REWRITE			X	
	DELETE			X	
	CLOSE	X	X	X	

Conceptos relacionados
“Organización de archivos indexados” en la página 128
“Organización de archivos relativa” en la página 129
“Acceso secuencial” en la página 129
“Acceso aleatorio” en la página 129
“Acceso dinámico” en la página 129

Lectura de registros de un archivo

Utilice la sentencia READ para recuperar registros de un archivo. Para leer un registro, debe haber abierto el archivo con OPEN INPUT o OPEN I-O (OPEN I-O no es válido para archivos secuenciales de línea). Compruebe la clave de estado de archivo después de cada READ.

Acerca de esta tarea

Puede recuperar registros en archivos secuenciales y secuenciales de línea sólo en la secuencia en la que se grabaron.

Puede recuperar registros en archivos de registros indexados y relativos de forma secuencial (según el orden ascendente de la clave para archivos indexados, o según las ubicaciones de registros relativos ascendentes para archivos relativos), de forma aleatoria o dinámica.

Al utilizar el acceso dinámico, puede conmutar entre leer registros secuencialmente y leer un registro específico directamente. Para la recuperación secuencial, codifique READ NEXT y READ PREVIOUS; y para la recuperación aleatoria (por clave), utilice READ.

Para leer secuencialmente a partir de un registro específico, utilice una sentencia START para establecer el indicador de posición de archivo para que apunte a un registro determinado antes de la sentencia READ NEXT o READ PREVIOUS . Si codifica START seguido de READ NEXT, se lee el siguiente registro y el indicador de posición de archivo se restablece en el siguiente registro. Si codifica START seguido de READ PREVIOUS, se lee el registro anterior y el indicador de posición de archivo se restablece en el registro anterior. Puede mover el indicador de posición de archivo aleatoriamente utilizando START, pero toda la lectura se realiza secuencialmente desde ese punto.

Puede continuar leyendo registros secuencialmente, o puede utilizar START para mover el indicador de posición de archivo. Por ejemplo:

```
START file-name KEY IS EQUAL TO ALTERNATE-RECORD-KEY
```

Si se realiza un READ directo para un archivo indexado basado en un índice alternativo para el que existen duplicados, sólo se recupera el primer registro del archivo (clúster base) con ese valor de clave alternativo. Necesita una serie de sentencias READ NEXT para recuperar cada uno de los registros que tienen la misma clave alternativa.Se devuelve un valor de estado de archivo de 02 si hay más registros con el mismo valor de clave alternativa que todavía se debe leer.Se devuelve un valor de 00 cuando se ha leído el último registro con ese valor de clave.

Conceptos relacionados
“Acceso secuencial” en la página 129
“Acceso aleatorio” en la página 129
“Acceso dinámico” en la página 129
“Organización de archivos y modalidad de acceso” en la página 127

Tareas relacionadas
“Apertura de un archivo” en la página 144
“Utilización de claves de estado de archivo” en la página 181

Referencias relacionadas
“Indicador de posición de archivo” en la página 144
cláusula FILE STATUS (*COBOL for Linux en x86 Consulta de lenguaje*)

Sentencias utilizadas al grabar registros en un archivo

La tabla siguiente muestra las sentencias COBOL que puede utilizar al crear o ampliar un archivo.

<table>
<tr><td colspan="5">Tabla 13. Sentencias utilizadas al grabar registros en un archivo</td></tr>
<tr><th>División</th><th>secuencial</th><th>Línea secuencial</th><th>Indexado</th><th>relativo</th></tr>
<tr><td rowspan="6">ENVIRONMENT</td><td colspan="4">SELECT</td></tr>
<tr><td colspan="4">ASSIGN</td></tr>
<tr><td>ORGANIZATION IS SEQUENTIAL</td><td>ORGANIZATION IS LINE SEQUENTIAL</td><td>ORGANIZATION IS INDEXED</td><td>ORGANIZATION IS RELATIVE</td></tr>
<tr><td colspan="2" rowspan="2">n/d</td><td>RECORD KEY</td><td rowspan="2">RELATIVE KEY</td></tr>
<tr><td>ALTERNATE RECORD KEY</td></tr>
<tr><td colspan="4">FILE STATUS</td></tr>
<tr><td colspan="4">ACCESS MODE</td></tr>
<tr><td>DATA</td><td colspan="4">Entrada FD</td></tr>
<tr><td rowspan="4">PROCEDURE</td><td colspan="4">OPEN OUTPUT</td></tr>
<tr><td colspan="4">OPEN EXTEND</td></tr>
<tr><td colspan="4">WRITE</td></tr>
<tr><td colspan="4">CLOSE</td></tr>
</table>

Conceptos relacionados
“Organización de archivos y modalidad de acceso” en la página 127

Tareas relacionadas
“Especificación de una organización de archivos y modalidad de acceso” en la página 127
“Apertura de un archivo” en la página 144
“Configuración de un campo para el estado de archivo” en la página 140
“Adición de registros a un archivo” en la página 148

Referencias relacionadas
“Sentencias PROCEDURE DIVISION utilizadas para actualizar archivos” en la página 150

Adición de registros a un archivo

La sentencia COBOL WRITE añade un registro a un archivo sin sustituir ningún registro existente. El registro que se va a añadir no debe ser mayor que el tamaño máximo de registro establecido cuando se definió el archivo. Compruebe la clave de estado de archivo después de cada sentencia WRITE .

Acerca de esta tarea

Adición secuencial de registros: Para añadir registros secuencialmente al final de un archivo que se ha abierto con OUTPUT o EXTEND, utilice ACCESS IS SEQUENTIAL y codifique la sentencia WRITE .

Los archivos secuenciales y secuenciales de línea siempre se escriben secuencialmente.

Para los archivos indexados, debe añadir nuevos registros en secuencia de teclas ascendente. Si se abre un archivo EXTEND, las claves de registro de los registros que se van a añadir deben ser superiores a la clave de registro primario más alta que había en el archivo cuando se abrió.

Para los archivos relativos, los registros deben estar en secuencia. Si codifica un elemento de datos RELATIVE KEY en la cláusula SELECT , el número de registro relativo del registro que se va a escribir se coloca en ese elemento de datos.

Adición aleatoria o dinámica de registros: cuando escribe registros en un archivo indexado para el que ha codificado ACCESS IS RANDOM o ACCESS IS DYNAMIC, puede escribir los registros en cualquier orden.

Conceptos relacionados
"Organización de archivos y modalidad de acceso" en la página 127

Tareas relacionadas
"Especificación de una organización de archivos y modalidad de acceso" en la página 127
"Utilización de claves de estado de archivo" en la página 181

Referencias relacionadas
"Sentencias utilizadas al grabar registros en un archivo" en la página 147
"Sentencias PROCEDURE DIVISION utilizadas para actualizar archivos" en la página 150

cláusula FILE STATUS (*COBOL for Linux en x86 Consulta de lenguaje*)

Sustitución de registros en un archivo

Para sustituir un registro en un archivo, utilice REWRITE si ha abierto el archivo para I-O. Si el archivo se ha abierto excepto para I-O, el registro no se sustituye y la clave de estado se establece en 49.

Acerca de esta tarea

Compruebe la clave de estado de archivo después de cada sentencia REWRITE .

Para archivos secuenciales, la longitud del registro de sustitución debe ser la misma que la longitud del registro original. Para archivos indexados o archivos relativos de longitud variable, puede cambiar la longitud del registro que sustituye.

Para sustituir un registro de forma aleatoria o dinámica, no tiene que READ primero el registro. En su lugar, localice el registro que desea sustituir como se indica a continuación:

- Para archivos indexados, mueva la clave de registro al elemento de datos RECORD KEY y, a continuación, utilice REWRITE.
- Para archivos relativos, mueva el número de registro relativo al elemento de datos RELATIVE KEY y, a continuación, utilice REWRITE.

Conceptos relacionados
"Organización de archivos y modalidad de acceso" en la página 127

Tareas relacionadas
"Apertura de un archivo" en la página 144
"Utilización de claves de estado de archivo" en la página 181

Referencias relacionadas
cláusula FILE STATUS (*COBOL for Linux en x86 Consulta de lenguaje*)

Supresión de registros de un archivo

Para eliminar un registro existente de un archivo indexado o relativo, abra el archivo como I-O y utilice la sentencia DELETE . No puede utilizar DELETE para un archivo secuencial o secuencial de línea.

Acerca de esta tarea

Si es ACCESS IS SEQUENTIAL, el programa COBOL debe leer primero el registro que se va a suprimir. La sentencia DELETE elimina el registro que se acaba de leer. Si la sentencia DELETE no va precedida de un READsatisfactorio, el registro no se suprime y la clave de estado del archivo se establece en 92.

Si ACCESS IS RANDOM o ACCESS IS DYNAMIC, no es necesario que el programa COBOL lea el registro que se va a suprimir. Para suprimir un registro, mueva la clave del registro al elemento de datos RECORD KEY y, a continuación, emita DELETE.

Compruebe la clave de estado de archivo después de cada sentencia DELETE .

Conceptos relacionados
“Organización de archivos y modalidad de acceso” en la página 127

Tareas relacionadas
“Apertura de un archivo” en la página 144
“Lectura de registros de un archivo” en la página 146
“Utilización de claves de estado de archivo” en la página 181

Referencias relacionadas
cláusula FILE STATUS (*COBOL for Linux en x86 Consulta de lenguaje*)

Sentencias PROCEDURE DIVISION utilizadas para actualizar archivos

La tabla siguiente muestra las sentencias que puede utilizar en PROCEDURE DIVISION para archivos secuenciales, secuenciales de línea, indexados y relativos.

Tabla 14. ***Sentencias PROCEDURE DIVISION utilizadas para actualizar archivos***

Método de acceso	secuencial	Línea secuencial	Indexado	relativo
ACCESS IS SEQUENTIAL	OPEN EXTEND WRITE CLOSE o OPEN I-O READ REWRITE CLOSE	OPEN EXTEND WRITE CLOSE	OPEN EXTEND WRITE CLOSE o OPEN I-O READ REWRITE DELETE CLOSE	OPEN EXTEND WRITE CLOSE o OPEN I-O READ REWRITE DELETE CLOSE
ACCESS IS RANDOM	No aplicable	No aplicable	OPEN I-O READ WRITE REWRITE DELETE CLOSE	OPEN I-O READ WRITE REWRITE DELETE CLOSE
ACCESS IS DYNAMIC (proceso secuencial)	No aplicable	No aplicable	OPEN I-O READ NEXT READ PREVIOUS START CLOSE	OPEN I-O READ NEXT READ PREVIOUS START CLOSE
ACCESS IS DYNAMIC (proceso aleatorio)	No aplicable	No aplicable	OPEN I-O READ WRITE REWRITE DELETE CLOSE	OPEN I-O READ WRITE REWRITE DELETE CLOSE

Conceptos relacionados
“Organización de archivos y modalidad de acceso” en la página 127

Tareas relacionadas
“Apertura de un archivo” en la página 144
“Lectura de registros de un archivo” en la página 146
“Adición de registros a un archivo” en la página 148
“Sustitución de registros en un archivo” en la página 149
“Supresión de registros de un archivo” en la página 149

Referencias relacionadas
“Sentencias utilizadas al grabar registros en un archivo” en la página 147

Utilización de archivos de Db2

Para acceder a los archivos Db2 desde una aplicación COBOL que se ejecuta bajo Linux, debe seguir las directrices para compilar y enlazar la aplicación y para identificar el sistema de archivos Db2 , la instancia y la base de datos de Db2 y los archivos Db2 .

Acerca de esta tarea

1. Compile y enlace los programas COBOL en la aplicación utilizando el mandato `cob2` .
2. Inicialice el entorno de Db2 ejecutando el perfil para la instancia de Db2 que desea utilizar.

   Por ejemplo, puede emitir el mandato siguiente para utilizar la instancia `db2inst1`:

   ```
   . /home/db2inst1/sqllib/db2profile
   ```

3. Asegúrese de que la instancia de Db2 se está ejecutando y de que puede conectarse a la base de datos a la que desea acceder.

   El ejemplo siguiente muestra el mandato db2 que puede utilizar para conectarse a la base de datos `db2cob`y muestra la respuesta del sistema:

   ```
   > db2 connect to db2cob
           Database Connection Information
           Database server        = DB2/Linux64 9.7.0
           SQL authorization ID   = MYUID
           Local database alias   = DB2COB
   ```

   Restricción: Tenga en cuenta que las aplicaciones COBOL pueden acceder a los archivos de Db2 en una sola base de datos e instancia de Db2 a la vez.
4. Establezca la variable de entorno DB2DBDFT en la base de datos deseada. Por ejemplo:

   ```
   export DB2DBDFT=db2cob
   ```

5. Para los archivos que utilizan Db2, especifique el sistema de archivos Db2 (ya sea como el valor de la opción de tiempo de ejecución `FILESYS` o directamente en el valor de nombre-asignación). Utilice el nombre de tabla completo de Db2 , incluido el nombre de esquema.

   Por ejemplo, el mandato siguiente completa la asignación de un archivo de transacción `TRANFILE` al nombre de archivo del sistema `TEST.TRANS` en el sistema de archivos Db2 :

   ```
   export TRANFILE=DB2-TEST.TRANS
   ```

Creación de archivos de Db2 :

Puede crear un archivo Db2 de varias maneras:

- Utilizando una sentencia 0PEN en el programa COBOL

- Utilizando el programa de utilidad TXSeries o CICS TX `cicsddt`

  Para obtener más información sobre el mandato `cicsddt` , consulte la documentación de TXSeries o CICS TX a la que se hace referencia a continuación.
- Utilizando el mandato `db2 create`

Por ejemplo, puede utilizar la siguiente secuencia de mandatos para crear un archivo relativo denominado EXAMPLE bajo el esquema `CICS`:

```
db2 create table cics.example\
"(rba char(4) not null for bit data, f1 varchar(80) not null for bit data)"
db2 create unique index cics.example0 on cics.example\
"(rba) disallow reverse scans"
db2 create unique index cics.example0@ on cics.example\
"(rba desc) disallow reverse scans"
```

Puede visualizar la tabla resultante emitiendo el mandato `db2 describe` . Por ejemplo:

```
> db2 describe table cics.example

                        Data type                     Column
Column name             schema    Data type name      Length     Scale Nulls
----------------------- --------- ------------------- ---------- ----- ------
RBA                     SYSIBM    CHARACTER                    4     0 No
F1                      SYSIBM    VARCHAR                     80     0 No

  2 record(s) selected.
```

El archivo `CICS.EXAMPLE` tiene registros de longitud variable que serían compatibles con una definición COBOL `FILE SECTION` de longitud de registro mínima entre 0 y 79.

Para obtener más información sobre las funciones que proporciona el programa de utilidad db2 , especifique el mandato db2.

Para obtener información sobre los requisitos adicionales que se aplican a la utilización de archivos Db2 con TXSeries o CICS TX, consulte la referencia relacionada sobre el sistema de archivos Db2 .

Tareas relacionadas
“Identificación de archivos” en la página 117
“Identificación de archivos de Db2” en la página 119
“Utilización de archivos y sentencias SQL de Db2 en el mismo programa” en la página 152
“Compilación desde la línea de mandatos” en la página 240
Capítulo 17, “Programación para un entorno Db2”, en la página 399

Referencias relacionadas
“Sistema de archivos Db2” en la página 123
“Variables de entorno de compilador y tiempo de ejecución” en la página 230
“ARCHIVOS” en la página 317
Documentación deTXSeries for Multiplatforms
Documentación de CICS TX

Utilización de archivos y sentencias SQL de Db2 en el mismo programa

Para utilizar sentencias EXEC SQL y E/S de archivo Db2 en el mismo programa, hay algunos hechos importantes que debe conocer, tal como se explica a continuación.

Acerca de esta tarea

- Ambos recursos (sentenciasEXEC SQL y E/S de archivo Db2) utilizan la *misma conexión* Db2 .
- Cada operación de actualización de E/S COBOL que utiliza el sistema de archivos Db2 se confirma en la base de datos inmediatamente.
- Si hay disponible una conexión Db2 existente, el sistema de archivos Db2 utiliza dicha conexión.

Si una conexión no está disponible, el sistema de archivos Db2 establece una conexión con la base de datos a la que hace referencia el valor de la variable de entorno DB2DBDFT .

Las sentencias EXEC SQL y la E/S de archivo Db2 pueden utilizar la misma base de datos o, si controla explícitamente la conexión, bases de datos diferentes.

Utilización de la misma base de datos:

Utilizar la misma base de datos para sentencias EXEC SQL y E/S de archivo Db2 en el mismo programa es más sencillo que utilizar bases de datos diferentes. Sin embargo, debe manejar esta configuración con cuidado:

- Para evitar anomalías, no se base en el uso del sistema de archivos Db2 de una conexión existente. Para garantizar resultados coherentes independientemente del tipo de acceso a la base de datos (Db2 file I/O o EXEC SQL) que se produzca en primer lugar, establezca la variable de entorno DB2DBDFT en la misma base de datos que utilizan las sentencias EXEC SQL .
- Db2 operaciones de actualización de archivo confirman todo el trabajo pendiente para la base de datos. Por lo tanto, retrotraiga o confirme las actualizaciones de EXEC SQL pendientes antes de iniciar cualquier operación de E/S de archivo de Db2 .

  Aunque la apertura de un archivo Db2 para entrada y lectura del archivo no provoca una confirmación de base de datos, se recomienda que no se base en este comportamiento.

Utilización de bases de datos diferentes:

Para utilizar bases de datos diferentes para sentencias EXEC SQL y para E/S de archivo de Db2 en el mismo programa, debe controlar explícitamente las conexiones de base de datos tal como se muestra en los ejemplos siguientes.

Supongamos que desea utilizar sentencias EXEC SQL con la base de datos `db2pli`y realizar la E/S de archivo de Db2 utilizando la base de datos `db2cob` estableciendo la variable de entorno DB2DBDFT:

```
export DB2DBDFT=db2cob
```

En el ejemplo siguiente, si se realiza la secuencia de pasos mostrados (con corchetes que indican pseudocódigo) no se utilizarán correctamente las bases de datos previstas, como explican los comentarios en línea:

```
<DB2 file I/O>                        *> Uses database db2cob (from DB2DBDFT)
EXEC SQL CONNECT TO db2pli END-EXEC   *> Switches to database db2pli
<Other EXEC SQL operations>           *> Use database db2pli
<DB2 file I/O>                        *> Uses the existing connection--and
. . .                                 *>   thus database db2pli--incorrectly!
```

Para acceder a las bases de datos previstas, primero desconéctese de la base de datos utilizada por las sentencias EXEC SQL antes de realizar cualquier operación de E/S de archivo de Db2 . A continuación, confíe en el valor de la variable de entorno DB2DBDFT o conéctese explícitamente a la base de datos que desea utilizar para la E/S del archivo Db2 .

La siguiente secuencia de pasos ilustra la dependencia de DB2DBDFT para realizar correctamente las conexiones previstas:

```
<DB2 file I/O>                        *> Uses database db2cob (from DB2DBDFT)
EXEC SQL CONNECT TO db2pli END-EXEC   *> Switches to database db2pli
<Other EXEC SQL operations>           *> Use database db2pli
* Commit or roll back pending operations
* here, because the following statement
* unconditionally commits pending work:
EXEC SQL CONNECT RESET END-EXEC       *> Disconnect from database db2pli
<DB2 file I/O>                        *> Uses database db2cob (from DB2DBDFT)
. . .
```

Tareas relacionadas

“Codificación de sentencias SQL” en la página 401

Referencias relacionadas
“Variables de entorno de compilador y tiempo de ejecución” en la página 230

Utilización de archivos MongoDB

Acerca de esta tarea

Para acceder a los archivos MongoDB desde una aplicación COBOL que se ejecuta en Linux, debe seguir este procedimiento para compilar y enlazar la aplicación y para identificar la base de datos y los archivos MongoDB .

Procedimiento

1. Compile y enlace los programas COBOL en la aplicación utilizando el mandato **cob2** .
2. Compruebe que el servidor MongoDB esté instalado y en ejecución. Para obtener detalles sobre la instalación y configuración de MongoDB, consulte la lista de referencias relacionadas. Tenga en cuenta que también tendrá que instalar el controlador C MongoDB disponible en mongoc.org tal como se documenta en los requisitos previos del sistema de la publicación IBM COBOL for Linux Installation Guide.
   a. Después de instalar MongoDB, debe haber un directorio para que la instancia de mongod almacene sus datos. Por ejemplo, `/var/lib/mongo`. Verifique que la propiedad es mongod y, si no es así, establézcala utilizando este mandato:

      ```
      chown -R mongod:mongod /var/lib/mongo
      ```

   b. MongoDB crea un archivo de socket para la conexión de base de datos en el directorio `/tmp` . Cambie la propiedad del archivo de socket a mongod utilizando el mandato siguiente:

      ```
      chown mongod:mongod mongodb-<port>.sock
      ```

      donde < *puerto*> es el puerto predeterminado de 27017, u otro valor elegido por el administrador de la base de datos.
   c. Inicie el servidor MongoDB utilizando el mandato siguiente:

      ```
      sudo systemctl start mongod
      ```

   d. Compruebe el estado del servidor MongoDB utilizando los mandatos siguientes:

      ```
      sudo systemctl status mongod
      ```

      Si el servidor se está ejecutando, debería ver una salida similar a la siguiente:

      ```
      mongod   587399     1 13 10:03  ?   00:00:00 /usr/bin/mongod -f /etc/mongod.conf
      ```

      Si el servidor no empieza a utilizar el mandato systemctl, inícielo manualmente utilizando el mandato:

      ```
      /usr/bin/mongod -f /etc/mongod.conf
      ```

3. MongoDB almacena información como objetos estructurados o no estructurados denominados documentos. Estos documentos se agrupan en colecciones. Un archivo MongoDB tiene un nombre de dos partes:

   ```
   [<database>.]<filename>.
   ```

   En términos de MongoDB , la base de datos es < *database*> y la colección es < *filename*>.
   a. Para identificar la base de datos MongoDB , establezca la variable de entorno VFS_MONGODB_DATABASE utilizando el mandato:

```
export VFS_MONGODB_DATABASE=<sampledb>
```

donde *samcomprob* es el nombre de la base de datos MongoDB .

b. Para comunicarse con un servidor MongoDB , necesita un URI de conexión. Establezca la variable de entorno VFS_MONGODB_URI, utilice el mandato siguiente:

```
export VFS_MONGODB_URI="mongodb://127.0.0.1:27017"
```

o

```
export VFS_MONGODB_URI="mongodb://localhost:27017"
```

donde 27017 es el puerto predeterminado. Si ha elegido un número de puerto diferente en el paso 2b, actualice VFS_MONGODB_URI a ese mismo número de puerto.

4. Establezca la variable de entorno COBRTOPT para indicar que está trabajando con archivos MongoDB :

```
export COBRTOPT=FILESYS=MONGO
```

5. Ejecute el programa COBOL. Las sentencias OPEN del programa abrirán el archivo en el servidor MongoDB .

   Cuando cierre el archivo, lo guardará en el servidor MongoDB .

6. Para mostrar el contenido de un archivo MongoDB o para eliminar un archivo MongoDB , utilice los programas de utilidad vfs_cat y vfs_rm. Por ejemplo:

```
vfs_cat mongo-<database>.<filename>
vfs_rm mongo-<database>.<filename>
```

   Tenga en cuenta que vfs_cat mostrará el archivo como si fuera un archivo secuencial, un orden de escritura ordenado (secuencial), un número de registro relativo o una clave primaria.

   De forma alternativa, si ha establecido la variable de entorno VFS_MONGODB_DATABASE en el nombre de base de datos MongoDB , puede utilizar los mandatos siguientes:

```
vfs_cat mongo-<filename>
vfs_rm mongo-<filename>
```

Qué hacer a continuación

Para obtener más información sobre Mongo, consulte las tareas y referencias relacionadas a continuación.

Tareas relacionadas

"Identificación de archivos" en la página 117
Para identificar un archivo, asocie el nombre de archivo que es interno del programa COBOL con el nombre de archivo del sistema correspondiente utilizando las cláusulas SELECT y ASSIGN en el párrafo FILE-CONTROL .

"Identificación de archivos MongoDB" en la página 121
Para identificar un archivo en el sistema de archivos MongoDB , especifique o de forma predeterminada el ID del sistema de archivos MONGO.

Referencias relacionadas

Requisito de software de MongoDB

"Sistema de archivos MongoDB" en la página 125
El sistema de archivos MongoDB admite archivos secuenciales, indexados y relativos.

"Variables de entorno de compilador y tiempo de ejecución" en la página 230
COBOL para Linux utiliza las variables de entorno siguientes que son comunes al compilador y al tiempo de ejecución.

"ARCHIVOS" en la página 317

FILESYS especifica el sistema de archivos que se va a utilizar para los archivos para los que no se ha especificado ningún sistema de archivos explícito mediante una cláusula ASSIGN o una variable de entorno. La opción se aplica a archivos secuenciales, relativos e indexados. No se aplica a los archivos secuenciales de línea, para los que el sistema de archivos debe especificarse como, o de forma predeterminada, LSQ (línea secuencial).

Referencias de MongoDB
Por qué utilizar MongoDB
¿Qué es una base de datos de documentos?

Utilización de archivos QSAM

Los archivos QSAM (método de acceso secuencial en cola) son archivos sin clave en los que los registros se colocan uno tras otro, según el orden de entrada.

Acerca de esta tarea

El programa puede procesar estos archivos sólo secuencialmente, recuperando registros utilizando la sentencia READ en el mismo orden en que están en el archivo. Cada registro se coloca después del registro anterior. Para procesar archivos QSAM en el programa, utilice sentencias de lenguaje COBOL que:

- Identifique y describa los archivos QSAM en ENVIRONMENT DIVISION y DATA DIVISION.
- Procese los registros en estos archivos en PROCEDURE DIVISION.

Después de crear un registro, no puede cambiar su longitud ni su posición en el archivo, y no puede suprimirlo.

Tareas relacionadas
"Identificación de archivos" en la página 117

Referencias relacionadas
"Sistema de archivos QSAM" en la página 124
"ARCHIVOS" en la página 317

Utilización de archivos SFS

Para acceder a los archivos SFS de CICS desde una aplicación COBOL que se ejecuta en Linux, debe seguir las directrices para compilar y enlazar y para identificar el sistema de archivos, el servidor SFS de CICS y los archivos SFS.

Acerca de esta tarea

Procedimiento

1. Compile y enlace los programas COBOL en la aplicación utilizando el mandato cob2 .
2. Asegúrese de que el servidor SFS CICS al que accederá la aplicación esté en ejecución.
3. (Opcional) Si la aplicación crea uno o más archivos SFS y desea asignar los archivos en un volumen de datos SFS que tiene un nombre distinto de sfs_SSFS_SERVER, puede especificar uno o ambos de los nombres siguientes:
   - El nombre del volumen de datos SFS en el que se van a crear los archivos SFS. Para ello, asigne un valor a la variable de entorno de ejecución CICS_SFS_INDEX_VOLUME. El volumen de datos debe haberse definido en el servidor SFS. Si no sabe qué volúmenes de datos están disponibles para el servidor SFS, emita el mandato `sfsadmin list lvols`.

     De forma predeterminada, los archivos SFS se crean en el volumen de datos denominado sfs_SSFS_SERVER.

- El nombre del volumen de datos SFS en el que se van a crear los archivos de índice alternativos, si los hay. Para ello, asigne un valor a la variable de entorno de ejecución CICS_SFS_INDEX_VOLUME. El volumen de datos debe haberse definido en el servidor SFS.

  De forma predeterminada, los archivos de índice alternativos se crean en el mismo volumen que los archivos base correspondientes.

4. Identifique cada archivo SFS:

   - Establezca el sistema de archivos predeterminado en SFS estableciendo la opción de tiempo de ejecución FILESYS de la siguiente manera:

     ```
     export COBRTOPT=FILESYS=SFS
     ```

     O bien, en un mandato `export` para cada archivo SFS, preceda el nombre de archivo y el nombre de servidor SFS con el ID de sistema de archivos SFS seguido de un guión (-), tal como se muestra a continuación.
   - El nombre del servidor SFS de CICS debe preceder al nombre de archivo.
   - Los nombres de archivo de índices alternativos Cualquiera deben empezar por el nombre de archivo base, seguido de un punto y coma (;) y el nombre de índice alternativo.

   Por ejemplo, if/.: /cics/sfs/*sfsServer* es el servidor SFS CICS , y SFS04A es un archivo SFS que tiene un índice alternativo SFS04A1, puede identificar SFS04A emitiendo el siguiente mandato `export` :

   ```
   export SFS04AEV="SFS-/.:/cics/sfs/sfsServer/SFS04A(/.:/cics/sfs/sfsServer/SFS04A;SFS04A1)"
   ```

Resultados

Para obtener más información sobre los nombres completos para servidores y archivos SFS, consulte la tarea relacionada sobre la identificación de archivos SFS CICS .

“Ejemplo: acceso a archivos SFS” en la página 157

Tareas relacionadas

“Identificación de archivos” en la página 117
“Identificación de archivos SFS” en la página 120
“Mejora del rendimiento de SFS” en la página 159

Referencias relacionadas

“Sistema de archivos SFS” en la página 126
“Variables de entorno de ejecución” en la página 234
“ARCHIVOS” en la página 317

Ejemplo: acceso a archivos SFS

El ejemplo siguiente muestra las descripciones de archivo COBOL que puede codificar y los mandatos `sfsadmin` y `export` que puede emitir para crear y acceder a dos archivos SFS.

SFS04 es un archivo indexado que no tiene ningún índice alternativo. SFS04A es un archivo indexado que tiene un índice alternativo, SFS04A1.

Descripciones de archivo COBOL

```
. . .
Environment division.
Input-output section.
File-control.
    select SFS04-file
```

```
        assign to SFS04EV
        access dynamic
        organization is indexed
        record key is SFS04-rec-num
        file status is SFS04-status.

    select SFS04A-file
        assign to SFS04AEV
        access dynamic
        organization is indexed
        record key is SFS04A-rec-num
        alternate record key is SFS04A-date-time
        file status is SFS04A-status.

Data division.
File section.
FD  SFS04-file.
  01 SFS04-record.
     05 SFS04-rec-num              pic x(10).
     05 SFS04-rec-data             pic x(70).
FD  SFS04A-file.
  01 SFS04A-record.
     05 SFS04A-rec-num             pic x(10).
     05 SFS04A-date-time.
        07 SFS04A-date-yyyymmdd    pic 9(8).
        07 SFS04A-time-hhmmsshh    pic 9(8).
        07 SFS04A-date-time-counter pic 9(8).
     05 SFS04A-rec-data            pic x(1000).
```

mandatos sfsadmin

Cree cada archivo indexado emitiendo el mandato `sfsadmin create clusteredfile` y añada un índice alternativo emitiendo el mandato `sfsadmin add index` :

```
sfsadmin create clusteredfile SFS04 2 \
PrimaryKey byteArray 10 \
DATA byteArray 70 \
primaryIndex -unique 1 PrimaryKey sfsVolume
#
sfsadmin create clusteredfile SFS04A 3 \
PrimaryKey byteArray 10 \
AltKey1 byteArray 24 \
DATA byteArray 1000 \
primaryIndex -unique 1 PrimaryKey sfsVolume
#
sfsadmin add index SFS04A SFS04A1 1 AltKey1
```

Tal como se muestra en el primer mandato `sfsadmin create clusteredfile` anterior, debe especificar los elementos siguientes:

- El nombre del nuevo archivo indexado (SFS04 en este ejemplo)
- El número de campos por registro (2)
- La descripción de cada campo (`PrimaryKey` y `DATA`, cada uno una matriz de bytes)
- El nombre del índice primario (`primaryIndex`)
- La opción `-unique`
- El número de campos en el índice primario (1)
- Los nombres de los campos en el índice primario (`PrimaryKey`)
- El nombre del volumen de datos en el que se va a almacenar el archivo (*`sfsVolume`*)

De forma predeterminada, CICS SFS permite claves duplicadas en el índice primario de un archivo en clúster. Sin embargo, debe especificar la opción `-unique` tal como se muestra anteriormente porque COBOL requiere que los valores de clave del índice primario sean exclusivos dentro del archivo.

Tal como se muestra en el mandato `sfsadmin add index` anterior, debe especificar los elementos siguientes:

- El nombre del archivo al que se va a añadir un índice alternativo (SFS04A en este ejemplo)
- El nombre del nuevo índice (SFS04A1)

- El número de campos que se van a utilizar como claves en el nuevo índice (1)
- Los nombres de los campos en el nuevo índice (`AltKey1`)

Para obtener detalles sobre la sintaxis de los mandatos `sfsadmin create clusteredfile` y `sfsadmin add index`, consulte las referencias relacionadas.

mandatos de exportación

Antes de ejecutar el programa que procesa los archivos SFS, emita estos mandatos `exportar` para especificar el vía de acceso (`/.:/cics/sfs`) al servidor SFS de CICS (*sfsServer*) que accederá a los archivos y el volumen de datos (*sfsVolume*) que almacenará los archivos:

```
# Set environment variables required by the SFS file system
# for SFS files:

export CICS_SFS_DATA_VOLUME=sfsVolume
export CICS_SFS_INDEX_VOLUME=sfsVolume

# Set SFS as the default file system:
export COBRTOPT=FILESYS=SFS

# Enable use of a short-form SFS specification:
export CICS_TK_SFS_SERVER=/.:/cics/sfs/sfsServer

# Set the environment variable to access SFS file SFS04
# (an example of using a short-form SFS specification):
export SFS04EV=SFS04

# Set the environment variable to access SFS
# file SFS04A and the alternate index SFS04A1:
```

```
export SFS04AEV="/.:/cics/sfs/sfsServer/SFS04A(/.:/cics/sfs/sfsServer/
SFS04A;SFS04A1)"
```

Referencias relacionadas
Documentación de TXSeries
Documentación de CICS TX

Mejora del rendimiento de SFS

Puede mejorar el rendimiento de las aplicaciones que acceden a los archivos SFS de dos maneras: utilizando el almacenamiento en memoria caché del lado del cliente en la máquina cliente y reduciendo la frecuencia de guardar los cambios en los archivos SFS.

Acerca de esta tarea

Tareas relacionadas
“Identificación de archivos SFS” en la página 120
“Habilitación del almacenamiento en memoria caché del lado del cliente” en la página 159
“Reducción de la frecuencia de guardar cambios” en la página 161
Mejora del rendimiento del SFS en la documentación de TXSeriesMemoria física y mejora del rendimiento en la documentación de CICS TX

Referencias relacionadas
“Sistema de archivos SFS” en la página 126

Habilitación del almacenamiento en memoria caché del lado del cliente

De forma predeterminada, los registros de los archivos SFS se graban y leen en el servidor SFS, y se necesita una llamada a procedimiento remoto (RPC) siempre que se acceda a un registro. Sin embargo, si habilita *almacenamiento en memoria caché del lado del cliente*, puede mejorar el rendimiento porque se necesita menos tiempo para acceder a los registros.

Acerca de esta tarea

Con el almacenamiento en memoria caché del lado del cliente, los registros se almacenan en la memoria local (una *memoria caché*) en la máquina cliente y se envían (*vaciados*) al servidor en un único RPC. Puede especificar uno o ambos de dos tipos de almacenamiento en memoria caché: *almacenamiento en memoria caché de lectura* y *almacenamiento en memoria caché de inserción*:

- Si habilita el almacenamiento en memoria caché de lectura, la primera lectura secuencial de un archivo hace que el registro actual y un número de registros vecinos se lean del servidor y se coloquen en la memoria caché de lectura. Las lecturas, actualizaciones y supresiones secuenciales posteriores se realizan en la memoria caché de lectura en lugar de en el servidor.
- Si habilita el almacenamiento en memoria caché de inserción, las operaciones de inserción (pero no las lecturas, actualizaciones o supresiones) se realizan en la memoria caché de inserción en lugar de en el servidor.

Para habilitar el almacenamiento en memoria caché del lado del cliente para todos los archivos SFS de la aplicación, establezca la variable de entorno CICS_VSAM_CACHE antes de ejecutar la aplicación. Para ver un diagrama de sintaxis que describe el establecimiento de CICS_VSAM_CACHE, consulte la referencia relacionada sobre las variables de entorno de ejecución.

Puede codificar un único valor para el tamaño de memoria caché para indicar que se va a utilizar el mismo número de páginas para la memoria caché de lectura y para la memoria caché de inserción, o puede codificar valores distintos para la memoria caché de lectura e inserción separando los valores por dos puntos (`:`). Unidades de tamaño Express como números de páginas. Si codifica cero como el tamaño de la memoria caché de lectura, la memoria caché de inserción, o ambos, ese tipo de almacenamiento en memoria caché está inhabilitado. Por ejemplo, el mandato siguiente establece el tamaño de la memoria caché de lectura en 16 páginas y el tamaño de la memoria caché de inserción en 64 páginas para cada archivo SFS de la aplicación:

```
export CICS_VSAM_CACHE=16:64
```

También puede especificar uno o ambos de los distintivos siguientes para que el almacenamiento en memoria caché del lado del cliente sea más flexible:

- Para permitir que se lean los datos no confirmados (registros que son nuevos o modificados pero que todavía no se han enviado al servidor, conocidos como registros *sucios*), especifique ALLOW_DIRTY_READS.

  Este distintivo elimina la restricción para el almacenamiento en memoria caché de lectura de que los archivos a los que se accede deben estar bloqueados.
- Para permitir que las inserciones se almacenen en memoria caché, especifique INSERTS_DESPITE_UNIQUE_INDICES.

  Este distintivo elimina la restricción para el almacenamiento en memoria caché de inserción de que todos los índices activos para archivos en clúster e índices alternativos activos para archivos de secuencia de entrada y relativos deben permitir duplicados.

Por ejemplo, el mandato siguiente permite la máxima flexibilidad:

```
export CICS_VSAM_CACHE=16:64:ALLOW_DIRTY_READS,INSERTS_DESPITE_UNIQUE_INDICES
```

Para establecer el almacenamiento en memoria caché del lado del cliente de forma diferente para determinados archivos, codifique una llamada putenv () que establezca CICS_VSAM_CACHE antes de la sentencia OPEN para cada archivo para el que desee cambiar la política de almacenamiento en memoria caché. Durante un programa, los valores de variable de entorno que realice en una llamada putenv () tienen prioridad sobre los valores de variable de entorno que realice en un mandato `export` .

“Ejemplo: establecimiento y acceso a variables de entorno” en la página 238

Tareas relacionadas

“Establecimiento de variables de entorno” en la página 229

Referencias relacionadas
“Sistema de archivos SFS” en la página 126
“Variables de entorno de ejecución” en la página 234

Reducción de la frecuencia de guardar cambios

Normalmente se produce un RPC para cada operación de grabación o actualización en un archivo SFS que no utiliza el almacenamiento en memoria caché del lado del cliente (es decir, la característica de *fuerza operativa* de SFS está habilitada). Todos los cambios de archivo que resultan de las operaciones de entrada-salida se confirman en disco antes de que el control vuelva a la aplicación.

Acerca de esta tarea

Puede cambiar este comportamiento para que el resultado de las operaciones de entrada-salida en los archivos SFS no se confirme en el disco hasta que se cierren los archivos (es decir, se utilice una estrategia *lazy-write*). Si cada cambio en un archivo SFS no se guarda inmediatamente, la aplicación puede ejecutarse más rápido.

Para cambiar el comportamiento de confirmación predeterminado para todos los archivos SFS de la aplicación, establezca la variable de entorno CICS_VSAM_AUTO_FLUSH en 0FF antes de ejecutar la aplicación:

```
export CICS_VSAM_AUTO_FLUSH=OFF
```

Para establecer el valor de vaciado de forma diferente para determinados archivos, codifique una llamada putenv () que establezca CICS_VSAM_AUTO_FLUSH antes de la sentencia 0PEN para cada archivo para el que desee cambiar el valor de vaciado. Durante un programa, los valores de variable de entorno que realice en una llamada putenv () tienen prioridad sobre los valores de variable de entorno que realice en un mandato `export` .

“Ejemplo: establecimiento y acceso a variables de entorno” en la página 238

Si el almacenamiento en memoria caché del lado del cliente está en vigor para un archivo SFS (es decir, la variable de entorno CICS_VSAM_CACHE está establecida en un valor distinto de cero válido), el valor de CICS_VSAM_AUTO_FLUSH para el archivo se ignora. La fuerza operativa está inhabilitada para ese archivo.

Tareas relacionadas
“Establecimiento de variables de entorno” en la página 229

Referencias relacionadas
“Sistema de archivos SFS” en la página 126
“Variables de entorno de ejecución” en la página 234

Capítulo 8. Ordenación y fusión de archivos

Acerca de esta tarea

Puede organizar los registros en una secuencia determinada utilizando una sentencia SORT o MERGE . Puede mezclar sentencias SORT y MERGE en el mismo programa COBOL.

SORT sentencia
Acepta la entrada (de un archivo o un procedimiento interno) que no está en secuencia y produce la salida (a un archivo o un procedimiento interno) en una secuencia solicitada. Puede añadir, suprimir o cambiar registros antes o después de que se ordenen.

MERGE sentencia
Compara registros de dos o más archivos secuenciados y los combina en orden. Puede añadir, suprimir o cambiar registros después de fusionarlos.

Un programa puede contener cualquier número de operaciones de clasificación y fusión. Pueden ser la misma operación realizada muchas veces o diferentes operaciones. Sin embargo, una operación debe finalizar antes de que comience otra.

Los pasos que realiza para ordenar o fusionar son generalmente los siguientes:

Procedimiento

1. Describa el archivo de clasificación o fusión que se va a utilizar para ordenar o fusionar.
2. Describa la entrada que se va a ordenar o fusionar. Si desea procesar los registros antes de ordenarlos, codifique un procedimiento de entrada.
3. Describa la salida de la ordenación o fusión. Si desea procesar los registros después de ordenarlos o fusionarlos, codifique un procedimiento de salida.
4. Solicite la clasificación o fusión.
5. Determine si la operación de clasificación o fusión ha sido satisfactoria.

Resultados

Conceptos relacionados
“Proceso de clasificación y fusión” en la página 163

Tareas relacionadas
“Descripción del archivo de clasificación o fusión” en la página 164
“Descripción de la entrada para ordenar o fusionar” en la página 165
“Descripción de la salida de la ordenación o fusión” en la página 167
“Solicitud de clasificación o fusión” en la página 168
“Determinación de si la clasificación o fusión se ha realizado correctamente” en la página 171
“Detención prematura de una operación de clasificación o fusión” en la página 175

Referencias relacionadas
Sentencia SORT (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia MERGE (*COBOL for Linux en x86 Consulta de lenguaje*)

Proceso de clasificación y fusión

Durante la ordenación de un archivo, todos los registros del archivo se ordenan de acuerdo con el contenido de uno o más campos (*claves*) de cada registro. Puede ordenar los registros en orden ascendente o descendente de cada clave.

Si hay varias claves, los registros se ordenan en primer lugar de acuerdo con el contenido de la primera clave (o primaria), a continuación, de acuerdo con el contenido de la segunda clave, y así sucesivamente.

Para ordenar un archivo, utilice la sentencia COBOL SORT.

Durante la fusión de dos o más archivos (que ya deben estar ordenados), los registros se combinan y ordenan de acuerdo con el contenido de una o más claves de cada registro. Puede ordenar los registros en orden ascendente o descendente de cada clave. Al igual que con la ordenación, los registros se ordenan primero de acuerdo con el contenido de la clave primaria, luego de acuerdo con el contenido de la segunda clave, y así sucesivamente.

Utilice `MERGE . . . USING` para nombrar los archivos que desea combinar en un archivo secuenciado. La operación de fusión compara las claves en los registros de los archivos de entrada y pasa los registros secuenciados uno a uno a la sentencia `RETURN` de un procedimiento de salida o al archivo que ha especificado en la frase `GIVING` .

Tareas relacionadas
"Establecimiento de criterios de ordenación o fusión" en la página 169

Referencias relacionadas
Sentencia SORT (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia MERGE (*COBOL for Linux en x86 Consulta de lenguaje*)

Descripción del archivo de clasificación o fusión

Describa el archivo de ordenación que se va a utilizar para ordenar o fusionar. Necesita cláusulas `SELECT` y entradas SD incluso si está ordenando o fusionando elementos de datos sólo desde `WORKING-STORAGE` o `LOCAL-STORAGE`.

Acerca de esta tarea

Codifique como se indica a continuación:

Procedimiento

1. Escriba una o más cláusulas `SELECT` en el párrafo `FILE-CONTROL` de `ENVIRONMENT DIVISION` para nombrar un archivo de ordenación. Por ejemplo:

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT Sort-Work-1 ASSIGN TO SortFile.
```

   Sort-Work-1 es el nombre del archivo en el programa. Utilice este nombre para hacer referencia al archivo.
2. Describa el archivo de ordenación en una entrada SD en el `FILE SECTION` de `DATA DIVISION`. Cada entrada SD debe contener una descripción de registro. Por ejemplo:

```
DATA DIVISION.
FILE SECTION.
SD  Sort-Work-1
    RECORD CONTAINS 100 CHARACTERS.
01  SORT-WORK-1-AREA.
    05  SORT-KEY-1   PIC  X(10).
    05  SORT-KEY-2   PIC  X(10).
    05  FILLER       PIC  X(80).
```

Resultados

El archivo descrito en una entrada SD es el archivo de trabajo utilizado para una operación de clasificación o fusión. No puede realizar ninguna operación de entrada o salida en este archivo.

Referencias relacionadas
“Entradas de FILE SECTION” en la página 10

Descripción de la entrada para ordenar o fusionar

Describa el archivo o archivos de entrada para ordenar o fusionar siguiendo el procedimiento siguiente.

Acerca de esta tarea

Procedimiento

1. Escriba una o más cláusulas SELECT en el párrafo FILE-CONTROL de ENVIRONMENT DIVISION para nombrar los archivos de entrada. Por ejemplo:

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT Input-File ASSIGN TO InFile.
```

   Archivo de entrada es el nombre del archivo en el programa. Utilice este nombre para hacer referencia al archivo.
2. Describa el archivo de entrada (o los archivos al fusionar) en una entrada FD en el FILE SECTION de DATA DIVISION. Por ejemplo:

```
DATA DIVISION.
FILE SECTION.
FD  Input-File
    RECORD CONTAINS 100 CHARACTERS.
01  Input-Record   PIC X(100).
```

Resultados

Tareas relacionadas
“Codificación del procedimiento de entrada” en la página 166
“Solicitud de clasificación o fusión” en la página 168

Referencias relacionadas
“Entradas de FILE SECTION” en la página 10

Ejemplo: descripción de los archivos de clasificación y entrada para SORT

El ejemplo siguiente muestra las entradas ENVIRONMENT DIVISION y DATA DIVISION necesarias para describir los archivos de trabajo de clasificación y un archivo de entrada.

```
 ID Division.
 Program-ID. SmplSort.
 Environment Division.
 Input-Output Section.
 File-Control.
*
* Assign name for a working file is treated as documentation.
*
     Select Sort-Work-1 Assign To SortFile.
     Select Sort-Work-2 Assign To SortFile.
     Select Input-File  Assign To InFile.
 . . .
 Data Division.
 File Section.
 SD  Sort-Work-1
     Record Contains 100 Characters.
 01  Sort-Work-1-Area.
     05  Sort-Key-1    Pic  X(10).
```

```
     05  Sort-Key-2    Pic  X(10).
     05  Filler        Pic  X(80).
 SD  Sort-Work-2
     Record Contains 30 Characters.
 01  Sort-Work-2-Area.
     05  Sort-Key      Pic  X(5).
     05  Filler        Pic  X(25).
 FD  Input-File
     Record Contains 100 Characters.
 01  Input-Record      Pic  X(100).
 . . .
 Working-Storage Section.
 01  EOS-Sw            Pic  X.
 01  Filler.
     05  Table-Entry Occurs 100 Times
             Indexed By X1    Pic X(30).
     . . .
```

Tareas relacionadas
"Solicitud de clasificación o fusión" en la página 168

Codificación del procedimiento de entrada

Para procesar los registros en un archivo de entrada antes de que se liberen al programa de ordenación, utilice la frase INPUT PROCEDURE de la sentencia SORT .

Acerca de esta tarea

Puede utilizar un procedimiento de entrada para:

- Libere elementos de datos en el archivo de clasificación de WORKING-STORAGE o LOCAL-STORAGE.
- Liberar registros que ya se han leído en otro lugar del programa.
- Leer registros de un archivo de entrada, seleccionarlos o procesarlos y liberarlos en el archivo de clasificación.

Cada procedimiento de entrada debe estar contenido en párrafos o secciones. Por ejemplo, para liberar registros de una tabla en WORKING-STORAGE o LOCAL-STORAGE en el archivo de ordenación SORT-WORK-2, puede codificar de la forma siguiente:

```
     SORT SORT-WORK-2
       ON ASCENDING KEY SORT-KEY
       INPUT PROCEDURE 600-SORT3-INPUT-PROC
     . . .
 600-SORT3-INPUT-PROC SECTION.
     PERFORM WITH TEST AFTER
       VARYING X1 FROM 1 BY 1 UNTIL X1 = 100
       RELEASE SORT-WORK-2-AREA FROM TABLE-ENTRY (X1)
     END-PERFORM.
```

Para transferir registros al programa de ordenación, todos los procedimientos de entrada deben contener al menos una sentencia RELEASE o RELEASE FROM . Para liberar A de X, por ejemplo, puede codificar:

```
MOVE X TO A.
RELEASE A.
```

De forma alternativa, puede codificar:

```
RELEASE A FROM X.
```

La tabla siguiente compara las sentencias RELEASE y RELEASE FROM .

RELEASE	RELEASE FROM

`MOVE EXT-RECORD`
`  TO SORT-EXT-RECORD`
`PERFORM RELEASE-SORT-RECORD`
`. . .`
`RELEASE-SORT-RECORD.`
`  RELEASE SORT-RECORD`
	`PERFORM RELEASE-SORT-RECORD`
	`. . .`
	`RELEASE-SORT-RECORD.`
	`  RELEASE SORT-RECORD`
	`  FROM SORT-EXT-RECORD`

Referencias relacionadas
“Restricciones en los procedimientos de entrada y salida” en la página 168
sentencia RELEASE (*COBOL for Linux en x86 Consulta de lenguaje*)

Descripción de la salida de la ordenación o fusión

Si la salida de la ordenación o fusión es un archivo, describa el archivo siguiendo el procedimiento siguiente.

Acerca de esta tarea

Procedimiento

1. Escriba una cláusula SELECT en el párrafo FILE-CONTROL de ENVIRONMENT DIVISION para nombrar el archivo de salida. Por ejemplo:

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT Output-File ASSIGN TO OutFile.
```

   Salida-Archivo es el nombre del archivo en el programa. Utilice este nombre para hacer referencia al archivo.
2. Describa el archivo de salida (o los archivos al fusionar) en una entrada FD en el FILE SECTION de DATA DIVISION. Por ejemplo:

```
DATA DIVISION.
FILE SECTION.
FD  Output-File
    RECORD CONTAINS 100 CHARACTERS.
01  Output-Record   PIC X(100).
```

Resultados

Tareas relacionadas
“Codificación del procedimiento de salida” en la página 167
“Solicitud de clasificación o fusión” en la página 168

Referencias relacionadas
“Entradas de FILE SECTION” en la página 10

Codificación del procedimiento de salida

Para seleccionar, editar o cambiar de otro modo los registros ordenados antes de escribirlos desde el archivo de trabajo de ordenación en otro archivo, utilice la frase OUTPUT PROCEDURE de la sentencia SORT .

Acerca de esta tarea

Cada procedimiento de salida debe estar contenido en una sección o en un párrafo. Un procedimiento de salida debe incluir los dos elementos siguientes:

- Al menos una sentencia RETURN o una sentencia RETURN con la frase INTO
- Cualquier sentencia necesaria para procesar los registros que la sentencia RETURN pone a disposición, de uno en uno,

La sentencia RETURN hace que cada registro ordenado esté disponible para el procedimiento de salida. (La sentencia RETURN para un archivo de clasificación es similar a una sentencia READ para un archivo de entrada.)

Puede utilizar las frases AT END y END-RETURN con la sentencia RETURN . Las sentencias imperativas de la frase AT END se realizan después de que se hayan devuelto todos los registros del archivo de ordenación. El terminador de ámbito explícito de END-RETURN delimita el ámbito de la sentencia RETURN .

Si utiliza RETURN INTO en lugar de RETURN, los registros se devolverán a WORKING-STORAGE, LOCAL-STORAGEo a un área de salida.

Referencias relacionadas
"Restricciones en los procedimientos de entrada y salida" en la página 168
sentencia RETURN (*COBOL for Linux en x86 Consulta de lenguaje*)

Restricciones en los procedimientos de entrada y salida

Se aplican varias restricciones a cada procedimiento de entrada o salida llamado por SORT y a cada procedimiento de salida llamado por MERGE.

Observe estas restricciones:

- El procedimiento no debe contener ninguna sentencia SORT o MERGE .
- Puede utilizar sentencias ALTER, GO TOy PERFORM en el procedimiento para hacer referencia a nombres de procedimiento fuera del procedimiento de entrada o salida. Sin embargo, el control debe volver al procedimiento de entrada o salida después de una sentencia GO TO o PERFORM .
- El resto de PROCEDURE DIVISION no debe contener ninguna transferencia de control a puntos dentro del procedimiento de entrada o salida (con la excepción de la devolución del control de una sección declarativa).
- En un procedimiento de entrada o salida, puede llamar a un programa. Sin embargo, el programa llamado no puede emitir una sentencia SORT o MERGE , y el programa llamado debe volver al llamante.
- Durante una operación SORT o MERGE , se utiliza el elemento de datos SD . No debe utilizarlo en el procedimiento de salida antes de que se ejecute el primer RETURN . Si mueve datos a esta área de registro antes de la primera sentencia RETURN , se sobrescribirá el primer registro que se devuelva.

Tareas relacionadas
"Codificación del procedimiento de entrada" en la página 166
"Codificación del procedimiento de salida" en la página 167

Solicitud de clasificación o fusión

Para leer registros de un archivo de entrada (archivos para MERGE) sin proceso preliminar, utilice SORT . . . USING o MERGE . . . USING y el nombre del archivo de entrada (archivos) que ha declarado en una cláusula SELECT .

Acerca de esta tarea

Para transferir registros ordenados o fusionados del programa de clasificación o fusión a otro archivo sin ningún proceso adicional, utilice SORT . . . GIVING o MERGE . . . GIVING y el nombre del archivo de salida que ha declarado en una cláusula SELECT . Por ejemplo:

```
SORT Sort-Work-1
    ON ASCENDING KEY Sort-Key-1
    USING Input-File
    GIVING Output-File.
```

Para SORT . . . USING o MERGE . . . USING, el compilador genera un procedimiento de entrada para abrir el archivo (archivos), leer los registros, liberar los registros para el programa de clasificación o fusión y cerrar el archivo (archivos). El archivo (archivos) no debe estar abierto cuando la sentencia SORT o MERGE empieza la ejecución. Para SORT . . . GIVING o MERGE . . . GIVING, el compilador genera un procedimiento de salida para abrir el archivo, devolver los registros, escribir los registros y cerrar el archivo. El archivo no debe estar abierto cuando la sentencia SORT o MERGE empieza la ejecución.

"Ejemplo: descripción de los archivos de clasificación y entrada para SORT" en la página 165

Si desea que se realice un procedimiento de entrada en los registros de clasificación antes de que se ordenen, utilice SORT . . . INPUT PROCEDURE. Si desea que se realice un procedimiento de salida en los registros ordenados, utilice SORT . . . OUTPUT PROCEDURE. Por ejemplo:

```
SORT Sort-Work-1
    ON ASCENDING KEY Sort-Key-1
    INPUT PROCEDURE EditInputRecords
    OUTPUT PROCEDURE FormatData.
```

"Ejemplo: clasificación con procedimientos de entrada y salida" en la página 171

Restricción: No puede utilizar un procedimiento de entrada con la sentencia MERGE . El origen de entrada para la operación de fusión debe ser una colección de archivos ya ordenados. Sin embargo, si desea que se realice un procedimiento de salida en los registros fusionados, utilice MERGE . . . OUTPUT PROCEDURE. Por ejemplo:

```
MERGE Merge-Work
    ON ASCENDING KEY Merge-Key
    USING Input-File-1 Input-File-2 Input-File-3
    OUTPUT PROCEDURE ProcessOutput.
```

En FILE SECTION, debe definir *Merge-Work* en una entrada de SD y los archivos de entrada en entradas de FD .

Referencias relacionadas
Sentencia SORT (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia MERGE (*COBOL for Linux en x86 Consulta de lenguaje*)

Establecimiento de criterios de ordenación o fusión

Para establecer criterios de ordenación o fusión, defina las claves en las que se va a realizar la operación.

Acerca de esta tarea

Siga estos pasos:

Procedimiento

1. En la descripción de registro de los archivos que se van a ordenar o fusionar, defina la clave o claves.

   Restricción: Una clave no se puede localizar de forma variable.

2. En la sentencia SORT o MERGE , especifique los campos clave que se van a utilizar para secuenciar codificando la frase ASCENDING o DESCENDING KEY , o ambas. Cuando codifica más de una clave, algunas pueden ser ascendentes y otras descendentes.

   Especifique los nombres de las claves en orden decreciente de significación. La clave más a la izquierda es la clave primaria. La siguiente clave es la clave secundaria, y así sucesivamente.

Resultados

Las claves SORT y MERGE pueden ser de clase alfabética, alfanumérica, nacional (si la opción de compilador NCOLLSEQ(BIN) está en vigor)o numérica (pero no numérica de USAGE NATIONAL). Si tiene USAGE NATIONAL, una clave puede ser de categoría nacional o puede ser un elemento de datos editado a nivel nacional o numérico. Una clave no puede ser un elemento de datos decimal nacional o un elemento de datos de coma flotante nacional.

El orden de clasificación de las claves nacionales viene determinado por el orden binario de las claves. Si especifica un elemento de datos nacional como clave, cualquier frase COLLATING SEQUENCE de la sentencia SORT o MERGE no se aplica a dicha clave.

Puede mezclar sentencias SORT y MERGE en el mismo programa COBOL. Un programa puede realizar cualquier número de operaciones de clasificación o fusión. Sin embargo, una operación debe finalizar antes de que pueda empezar otra.

Tareas relacionadas
"Control de la secuencia de clasificación con un entorno local" en la página 219

Referencias relacionadas
"NCOLLSEQ" en la página 290
Sentencia SORT (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia MERGE (*COBOL for Linux en x86 Consulta de lenguaje*)

Elección de secuencias de clasificación alternativas

Puede ordenar o fusionar registros en una secuencia de clasificación que especifique para las claves de caracteres de un solo byte. La secuencia de clasificación predeterminada es la secuencia de clasificación especificada por el valor de entorno local en vigor en tiempo de compilación a menos que codifique la cláusula PROGRAM COLLATING SEQUENCE en el párrafo OBJECT-COMPUTER .

Acerca de esta tarea

Para alterar temporalmente la secuencia predeterminada, utilice la frase COLLATING SEQUENCE de la sentencia SORT o MERGE . Puede utilizar secuencias de clasificación diferentes para cada sentencia SORT o MERGE del programa.

La cláusula PROGRAM COLLATING SEQUENCE y la frase COLLATING SEQUENCE sólo se aplican a las claves de clase alfabética o alfanumérica. La frase COLLATING SEQUENCE sólo es válida cuando está en vigor una página de códigos ASCII de un solo byte.

Tareas relacionadas
"Especificación del orden de clasificación" en la página 6
"Control de la secuencia de clasificación con un entorno local" en la página 219
"Establecimiento de criterios de ordenación o fusión" en la página 169

Referencias relacionadas
OBJECT-párrafo COMPUTER (*COBOL for Linux en x86 Consulta de lenguaje*)
Sentencia SORT (*COBOL for Linux en x86 Consulta de lenguaje*)
Clases y categorías de datos (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: clasificación con procedimientos de entrada y salida

El ejemplo siguiente muestra el uso de un procedimiento de entrada y salida en una sentencia SORT . El ejemplo también muestra cómo puede definir una clave primaria (SORT-GRID-LOCATION) y una clave secundaria (SORT-SHIFT) antes de utilizarlas en la sentencia SORT .

```
DATA DIVISION.
. . .
SD  SORT-FILE
    RECORD CONTAINS 115 CHARACTERS
    DATA RECORD SORT-RECORD.
01  SORT-RECORD.
    05  SORT-KEY.
        10  SORT-SHIFT              PIC X(1).
        10  SORT-GRID-LOCATION      PIC X(2).
        10  SORT-REPORT             PIC X(3).
    05  SORT-EXT-RECORD.
        10  SORT-EXT-EMPLOYEE-NUM   PIC X(6).
        10  SORT-EXT-NAME           PIC X(30).
        10  FILLER                  PIC X(73).
. . .
WORKING-STORAGE SECTION.
01  TAB1.
    05 TAB-ENTRY OCCURS 10 TIMES
           INDEXED BY TAB-INDX.
        10  WS-SHIFT                PIC X(1).
        10  WS-GRID-LOCATION        PIC X(2).
        10  WS-REPORT               PIC X(3).
        10  WS-EXT-EMPLOYEE-NUM     PIC X(6).
        10  WS-EXT-NAME             PIC X(30).
        10  FILLER                  PIC X(73).
. . .
PROCEDURE DIVISION.
    . . .
*****************************************************************************
* This SORT statement will do a 'table sort' using Format 1 SORT, and will  *
* sort records in SORT-FILE in an ascending sequence based on the data      *
* in sort keys SORT-GRID-LOCATION (primary) and SORT-SHIFT (secondary).     *
* The source of the file records is data in table TAB-ENTRY                 *
* which is acquired in the input procedure 600-SORT3-INPUT and then         *
* output back to the table in the output procedure 700-SORT3-OUTPUT.        *
*****************************************************************************
    SORT SORT-FILE
        ON ASCENDING KEY SORT-GRID-LOCATION SORT-SHIFT
        INPUT PROCEDURE 600-SORT3-INPUT
        OUTPUT PROCEDURE 700-SORT3-OUTPUT.
    . . .
600-SORT3-INPUT.
    PERFORM VARYING TAB-INDX FROM 1 BY 1 UNTIL TAB-INDX > 10
        RELEASE SORT-RECORD FROM TAB-ENTRY(TAB-INDX)
    END-PERFORM.
. . .
700-SORT3-OUTPUT.
    PERFORM VARYING TAB-INDX FROM 1 BY 1 UNTIL TAB-INDX > 10
        RETURN SORT-FILE INTO TAB-ENTRY(TAB-INDX)
            AT END DISPLAY 'Out Of Records In SORT File'
        END-RETURN
    END-PERFORM.
```

Tareas relacionadas

“Solicitud de clasificación o fusión” en la página 168

Determinación de si la clasificación o fusión se ha realizado correctamente

La sentencia SORT o MERGE devuelve un código de terminación de 0 (finalización satisfactoria) o 16 (finalización no satisfactoria) después de que haya finalizado cada clasificación o fusión. El código de terminación se almacena en el registro especial SORT-RETURN .

Acerca de esta tarea

Debe probar si se ha completado correctamente después de cada sentencia SORT o MERGE . Por ejemplo:

```
    SORT SORT-WORK-2
        ON ASCENDING KEY SORT-KEY
        INPUT PROCEDURE IS 600-SORT3-INPUT-PROC
        OUTPUT PROCEDURE IS 700-SORT3-OUTPUT-PROC.
    IF SORT-RETURN NOT=0
        DISPLAY "SORT ENDED ABNORMALLY. SORT-RETURN = " SORT-RETURN.
    . . .
600-SORT3-INPUT-PROC SECTION.
    . . .
700-SORT3-OUTPUT-PROC SECTION.
    . . .
```

Si no hace referencia a SORT-RETURN en ningún lugar del programa, el tiempo de ejecución COBOL prueba el código de finalización. Si es 16, COBOL emite un mensaje de diagnóstico de tiempo de ejecución y termina la unidad de ejecución (o la hebra, en un entorno multihebra). El mensaje de diagnóstico contiene un número de error de clasificación o fusión que puede ayudarle a determinar la causa del problema.

Si prueba SORT-RETURN para una o más sentencias SORT o MERGE (pero no necesariamente todas), el tiempo de ejecución COBOL no comprueba el código de finalización. Sin embargo, puede obtener el número de error de clasificación o fusión después de cualquier sentencia SORT o MERGE llamando al servicio iwzGetSortErrno; por ejemplo:

```
77  sortErrno    PIC 9(9)   COMP-5.
. . .
    CALL 'iwzGetSortErrno'  USING  sortErrno
. . .
```

Consulte la referencia relacionada a continuación para obtener una lista de los números de error y sus significados.

Referencias relacionadas
“Ordenar y fusionar números de error” en la página 172

Ordenar y fusionar números de error

Si no hace referencia a SORT-RETURN en el programa y el código de finalización de una operación de clasificación o fusión es 16, COBOL para Linux emite un mensaje de diagnóstico de tiempo de ejecución que contiene uno de los números de error distintos de cero que se muestran en la tabla siguiente.

Tabla 15. ***Ordenar y fusionar números de error***

Número de error	**Descripción**
0	Sin error
1	El registro está desordenado.
2	Se han detectado registros con la misma clave.
3	Se han especificado varias funciones principales (error interno).
4	Error en el archivo de parámetros
5	No se ha podido abrir el archivo de parámetros.
6	Falta el operando en la opción
7	Falta el operando en la opción ampliada

*Tabla 15. **Ordenar y fusionar números de error** (continuación)*	
Número de error	**Descripción**
8	Operando no válido en la opción
9	Operando no válido en la opción ampliada
10	Se ha especificado una opción no válida.
11	Se ha especificado una opción ampliada no válida.
12	Se ha especificado un directorio temporal no válido.
13	Se ha especificado un nombre de archivo no válido.
14	Se ha especificado un campo no válido.
15	Falta un campo en el registro.
16	Un campo era demasiado corto en el registro.
17	Error de sintaxis en la especificación SELECT
18	Se ha especificado una constante no válida en SELECT.
19	Comparación no válida entre constante y tipo de datos en SELECT
20	Comparación no válida entre dos tipos de datos en SELECT
21	Error de sintaxis en especificación de formato
22	Error de sintaxis en la especificación de reformato
23	Se ha especificado una constante no válida en la especificación de reformato.
24	Error de sintaxis en la especificación de suma
25	Se ha especificado un distintivo varias veces.
26	Se han especificado demasiadas salidas.
27	No se ha especificado ningún origen de entrada.
28	No se ha especificado ningún destino de salida.
29	Se ha especificado un modificador no válido.
30	La suma no está permitida.
31	El registro es demasiado corto.
32	El registro es demasiado largo.
33	Se ha detectado un campo empaquetado o con zona no válido.
34	Error de lectura en archivo
35	Error de grabación en el archivo
36	No se puede abrir el archivo de entrada.
37	No se puede abrir el archivo de mensajes.
38	Error de archivo SdU o SFS
39	Espacio insuficiente en almacenamientos intermedios de destino
40	No hay suficiente espacio de disco temporal
41	No hay espacio suficiente para el archivo de salida

*Tabla 15. **Ordenar y fusionar números de error** (continuación)*	
Número de error	**Descripción**
42	Se ha atrapado una señal inesperada.
43	Se ha devuelto un error de la salida de entrada.
44	La salida de salida ha devuelto un error.
45	Se han devuelto datos inesperados de la salida de usuario de salida.
46	Se ha devuelto un valor utilizado de bytes no válido de la salida de entrada.
47	Se ha devuelto un valor de bytes utilizados no válido desde la salida de salida.
48	SMARTsort no está activo.
49	Almacenamiento insuficiente para continuar la ejecución
50	El archivo de parámetros era demasiado grande.
51	Comillas simples no coincidentes
52	Comillas no coincidentes
53	Se han especificado opciones en conflicto.
54	El campo de longitud del registro no es válido.
55	El último campo del registro no es válido.
56	No se ha especificado el formato de registro necesario.
57	No se puede abrir el archivo de salida.
58	No se puede abrir el archivo temporal.
59	Organización de archivo no válida
60	La salida de usuario no está soportada con la organización de archivos especificada.
61	El entorno local no es conocido en el sistema.
62	El registro contiene un carácter multibyte no válido.
63	El archivo no era SdU ni SFS.
64	No se puede utilizar ninguna clave especificada en S0RT para la definición del archivo de salida indexado.
65	La longitud de registro para un archivo SdU o SFS no era correcta.
66	La creación del archivo de opciones SMARTsort ha fallado.
67	Debe especificarse un nombre de vía de acceso completo y no relativo como directorio de trabajo.
68	Debe especificarse una opción necesaria.
69	El nombre de vía de acceso no es válido.
79	Se ha alcanzado el número máximo de archivos temporales.
501	Función no válida
502	Tipo de registro no válido
503	Longitud de registro no válida

*Tabla 15. **Ordenar y fusionar números de error** (continuación)*	
Número de error	**Descripción**
504	Error de longitud de tipo
505	Tipo no válido
506	Número de claves no coincidentes
507	El tipo es demasiado largo.
508	Desplazamiento de clave no válido
509	Clave ascendente o descendente no válida
510	Claves solapadas no válidas
511	No se ha definido ninguna clave.
512	No se ha especificado ningún archivo de entrada.
513	No se ha especificado ningún archivo de salida.
514	Archivos de entrada de tipo mixto
515	Archivos de salida de tipo mixto
516	Almacenamiento intermedio de trabajo de entrada no válido
517	Almacenamiento intermedio de trabajo de salida no válido
518	Error de E/S de entrada COBOL
519	Error de E/S de salida COBOL
520	Función no soportada
521	Clave no válida
522	Archivo `USING` no válido
523	Archivo `GIVING` no válido
524	No se ha proporcionado ningún directorio de trabajo.
525	El directorio de trabajo no existe.
526	No se ha asignado la clasificación común.
527	No hay almacenamiento para clasificación común
528	No se ha asignado el almacenamiento intermedio binario.
529	No se ha asignado el almacenamiento intermedio de archivo secuencial de línea.
530	La asignación de espacio de trabajo ha fallado.
531	La asignación de FCB ha fallado.

Detención prematura de una operación de clasificación o fusión

Para detener una operación de clasificación o fusión, mueva el entero 16 al registro especial `SORT-RETURN` .

Acerca de esta tarea

Mueva 16 al registro de una de las siguientes maneras:

- Utilice MOVE en un procedimiento de entrada o salida.

  El proceso de clasificación o fusión se detendrá inmediatamente después de que se realice la siguiente sentencia RELEASE o RETURN .
- Restablezca el registro en una sección declarativa especificada durante el proceso de un archivo USING o GIVING .

  El proceso de clasificación o fusión se detendrá al salir de la sección declarativa.

A continuación, el control vuelve a la sentencia que sigue a la sentencia SORT o MERGE .

Capítulo 9. manejar errores

Acerca de esta tarea

Coloque código en los programas que anticipa posibles problemas del sistema o del tiempo de ejecución. Si no incluye dicho código, los datos de salida o los archivos podrían estar dañados, y el usuario podría ni siquiera ser consciente de que hay un problema.

El código de manejo de errores puede realizar acciones como, por ejemplo, manejar la situación, emitir un mensaje o detener el programa. Por ejemplo, puede crear rutinas de detección de errores para errores de entrada de datos o para errores tal como los define la instalación. En cualquier caso, codificar un mensaje de advertencia es una buena idea.

COBOL para Linux contiene elementos especiales para ayudarle a anticipar y corregir las condiciones de error:

- `ON OVERFLOW` en operaciones `STRING` y `UNSTRING`
- `ON SIZE ERROR` en operaciones aritméticas
- Elementos para manejar errores de entrada o salida
- `ON EXCEPTION` o `ON OVERFLOW` en sentencias `CALL`

Tareas relacionadas
“Manejo de errores al unir y dividir series” en la página 177
“Manejo de errores en operaciones aritméticas” en la página 178
“Manejo de errores en operaciones de entrada y salida” en la página 178
“Manejo de errores al llamar a programas” en la página 185

Manejo de errores al unir y dividir series

Durante la unión o división de series, el puntero utilizado por `STRING` o `UNSTRING` puede estar fuera del rango del campo de recepción. Existe una condición de desbordamiento potencial, pero COBOL no deja que se produzca el desbordamiento.

Acerca de esta tarea

En su lugar, la operación `STRING` o `UNSTRING` no se completa, el campo de recepción permanece sin cambios y el control pasa a la siguiente sentencia secuencial. Si no codifica la frase `ON OVERFLOW` de la sentencia `STRING` o `UNSTRING` , no se le notificará de la operación incompleta.

Tenga en cuenta la sentencia siguiente:

```
String Item-1 space Item-2 delimited by Item-3
    into Item-4
    with pointer String-ptr
    on overflow
        Display "A string overflow occurred"
End-String
```

Estos son los valores de datos antes y después de realizar la sentencia:

Elemento de datos	PICTURE	Valor antes	Valor después de
Item-1	X(5)	AAAAA	AAAAA
Item-2	X(5)	EEEAA	EEEAA
Item-3	X(2)	ES	ES

Elemento de datos	PICTURE	Valor antes	Valor después de
Item-4	X(8)	*bbbbbbbb*[1]	*bbbbbbbb*[1]
String-ptr	9(2)	0	0
1. El símbolo *b* representa un espacio en blanco.			

Puesto que String-ptr tiene un valor (0) que no alcanza el campo de recepción, se produce una condición de desbordamiento y la operación STRING no se ha completado. (El desbordamiento también se produciría si String-ptr fuera mayor que 9.) Si no se hubiera especificado ON OVERFLOW , no se le notificaría que el contenido de Item-4 permanece sin cambios.

Manejo de errores en operaciones aritméticas

Los resultados de las operaciones aritméticas pueden ser mayores que el campo de punto fijo que las va a contener, o puede que haya intentado dividir por cero. En cualquier caso, la cláusula ON SIZE ERROR después de la sentencia ADD, SUBTRACT, MULTIPLY, DIVIDEo COMPUTE puede manejar la situación.

Acerca de esta tarea

Para que ON SIZE ERROR funcione correctamente para el desbordamiento de coma fija y el desbordamiento decimal, debe especificar la opción de tiempo de ejecución TRAP(ON) .

Se realizará la sentencia imperativa de la cláusula ON SIZE ERROR y el campo de resultado no cambiará en estos casos:

- Desbordamiento de punto fijo
- División por cero
- Cero elevado a la potencia cero
- Cero elevado a un número negativo
- Número negativo elevado a una potencia fraccional

Ejemplo: comprobación de división por cero

El ejemplo siguiente muestra cómo puede codificar una sentencia imperativa ON SIZE ERROR para que el programa emita un mensaje informativo si se produce una división por cero.

```
DIVIDE-TOTAL-COST.
    DIVIDE TOTAL-COST BY NUMBER-PURCHASED
        GIVING ANSWER
        ON SIZE ERROR
          DISPLAY "ERROR IN DIVIDE-TOTAL-COST PARAGRAPH"
          DISPLAY "SPENT " TOTAL-COST, " FOR " NUMBER-PURCHASED
          PERFORM FINISH
    END-DIVIDE
    . . .
    FINISH.
    STOP RUN.
```

Si se produce una división por cero, el programa escribe un mensaje y detiene la ejecución del programa.

Manejo de errores en operaciones de entrada y salida

Cuando una operación de entrada o salida falla, COBOL no realiza automáticamente la acción correctiva. Puede elegir si el programa continuará ejecutándose después de un error de entrada o salida menor que grave.

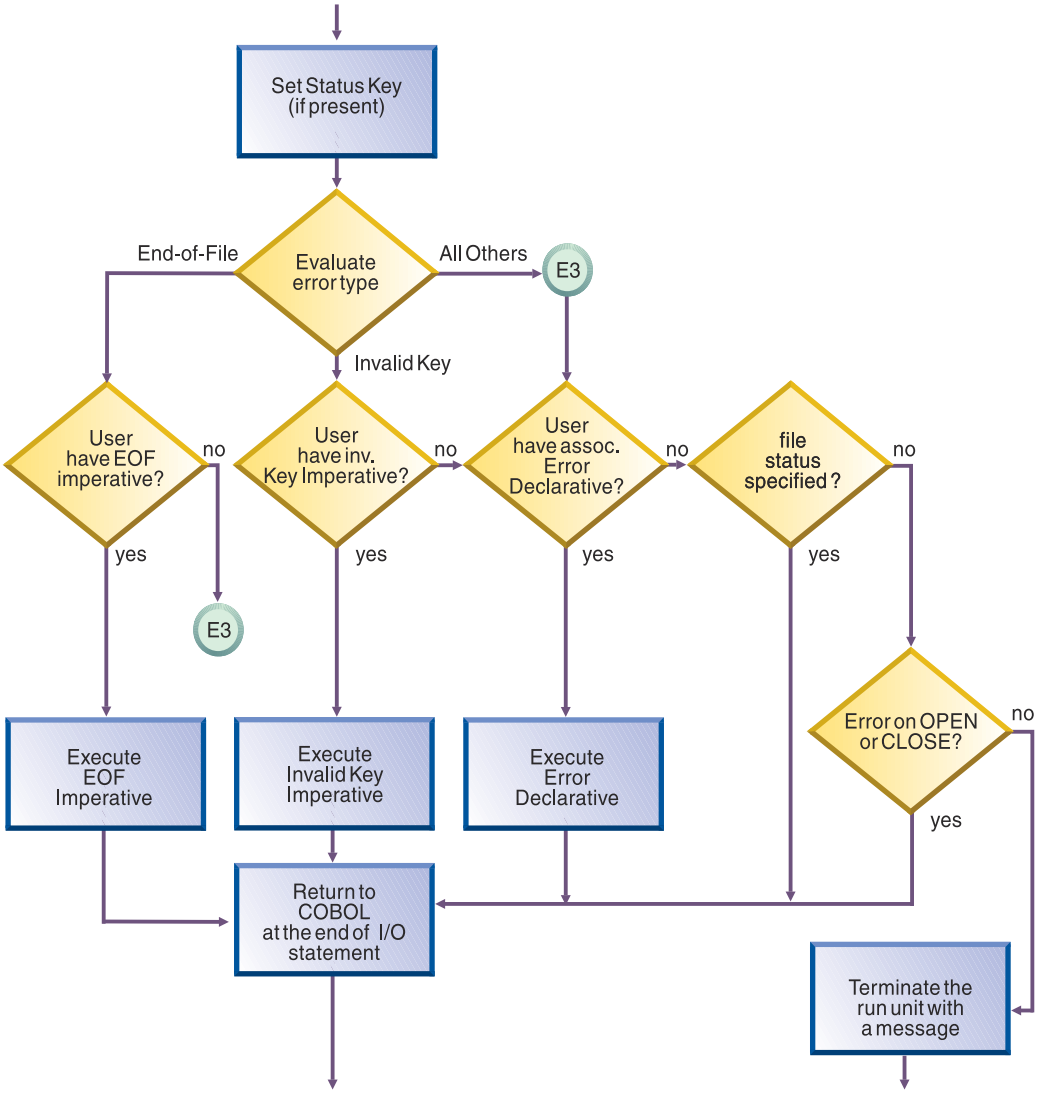
Acerca de esta tarea

Puede utilizar cualquiera de las técnicas siguientes para interceptar y manejar determinadas condiciones o errores de entrada o salida:

- Condición de fin de archivo (`AT END`)
- Declaraciones de `ERROR`
- Cláusula `FILE STATUS` y clave de estado de archivo
- Código de estado del sistema de archivos
- Frases de declaración imperativa en sentencias `READ` o `WRITE`
- Frase `INVALID KEY`

Para que el programa continúe, debe codificar el procedimiento de recuperación de errores adecuado. Puede codificar, por ejemplo, un procedimiento para comprobar el valor de la clave de estado de archivo. Si no maneja un error de entrada o salida de ninguna de estas maneras, se escribe un mensaje de tiempo de ejecución COBOL y la unidad de ejecución finaliza.

La figura siguiente muestra el flujo de lógica después de un error de entrada o salida del sistema de archivos:



Tareas relacionadas

“Apertura de archivos opcionales” en la página 140
“Utilización de la condición de fin de archivo (AT END)” en la página 180

“Declaración de ERROR de codificación” en la página 180
“Utilización de claves de estado de archivo” en la página 181
“Utilización de códigos de estado del sistema de archivos” en la página 182
“Codificación de frases INVALID KEY” en la página 184

Referencias relacionadas
Clave de estado de archivo (*COBOL for Linux en x86 Consulta de lenguaje*)

Utilización de la condición de fin de archivo (AT END)

La frase AT END de la sentencia READ se codifica para manejar errores o condiciones normales, según el diseño del programa. Al final del archivo, se realiza la frase AT END . Si no codifica una frase AT END , se realiza la declarativa ERROR asociada.

Acerca de esta tarea

En muchos diseños, la lectura secuencial hasta el final de un archivo se realiza intencionadamente y se espera la condición AT END . Por ejemplo, supongamos que está procesando un archivo que contiene transacciones para actualizar un archivo principal :

```
PERFORM UNTIL TRANSACTION-EOF = "TRUE"
  READ UPDATE-TRANSACTION-FILE INTO WS-TRANSACTION-RECORD
    AT END
      DISPLAY "END OF TRANSACTION UPDATE FILE REACHED"
      MOVE "TRUE" TO TRANSACTION-EOF
  END READ
  . . .
END-PERFORM
```

Cualquier frase NOT AT END sólo se realiza si la sentencia READ se completa correctamente. Si la operación READ falla debido a una condición distinta de fin de archivo, no se realiza ni la frase AT END ni la frase NOT AT END . En su lugar, el control pasa al final de la sentencia READ después de realizar cualquier procedimiento declarativo asociado.

Puede elegir no codificar una frase AT END o un procedimiento declarativo EXCEPTION , sino codificar una cláusula de clave de estado para el archivo en su lugar. En ese caso, el control pasa a la siguiente instrucción secuencial después de la sentencia de entrada o salida que ha detectado la condición de fin de archivo. En ese lugar, tenga algún código que tome las medidas adecuadas.

Referencias relacionadas
Frases AT END (*COBOL for Linux en x86 Consulta de lenguaje*)

Declaración de ERROR de codificación

Puede codificar uno o varios procedimientos declarativos de ERROR a los que se dará control si se produce un error de entrada o salida durante la ejecución del programa. Si no codifica dichos procedimientos, el trabajo podría cancelarse o terminarse de forma anómala después de que se produzca un error de entrada o salida.

Acerca de esta tarea

Coloque cada uno de estos procedimientos en la sección de declaraciones de PROCEDURE DIVISION. Puede codificar:

- Un único procedimiento común para todo el programa
- Procedimientos para cada modalidad de apertura de archivo (si INPUT, OUTPUT, I-Oo EXTEND)
- Procedimientos individuales para cada archivo

En un procedimiento declarativo de ERROR , puede codificar la acción correctiva, reintentar la operación, continuar o finalizar la ejecución. (Sin embargo, si continúa procesando un archivo bloqueado, es posible que pierda los registros restantes en un bloque después del registro que ha causado el error.) Puede

utilizar el procedimiento declarativo ERROR en combinación con la clave de estado de archivo si desea un análisis adicional del error.

Resultados

Referencias relacionadas
EXCEPTION/ERROR declarativo (*COBOL for Linux en x86 Consulta de lenguaje*)

Utilización de claves de estado de archivo

Después de realizar cada sentencia de entrada o salida en un archivo, el sistema actualiza los valores en las posiciones de dos dígitos de la clave de estado del archivo. En general, un cero en la primera posición indica una operación satisfactoria, y un cero en ambas posiciones significa que no se ha producido nada anómalo.

Acerca de esta tarea

Establezca una clave de estado de archivo codificando:

- La cláusula `FILE STATUS` en el párrafo `FILE-CONTROL` :

```
FILE STATUS IS data-name-1
```

- Definiciones de datos en `DATA DIVISION` (`WORKING-STORAGE`, `LOCAL-STORAGE`o `LINKAGE SECTION`), por ejemplo:

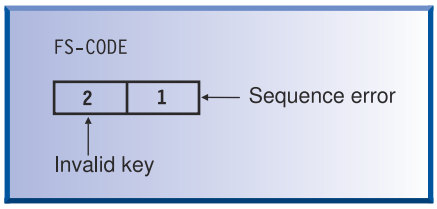
```
WORKING-STORAGE SECTION.
01  data-name-1  PIC 9(2)  USAGE NATIONAL.
```

Especifique la clave de estado de archivo *data-name-1* como un elemento nacional de categoría o alfanumérico de dos caracteres, o como un elemento decimal con zona o decimal nacional de dos dígitos. Este *data-name-1* no se puede localizar de forma variable.

El programa puede comprobar la clave de estado del archivo para descubrir si se ha producido un error y, si es así, qué tipo de error se ha producido. Por ejemplo, supongamos que una cláusula `FILE STATUS` está codificada de este modo:

```
FILE STATUS IS FS-CODE
```

`FS-CODE` es utilizado por COBOL para contener información de estado como la siguiente:



Siga estas reglas para cada archivo:

- Defina una clave de estado de archivo diferente para cada archivo.

  Hacerlo significa que puede determinar la causa de una excepción de entrada o salida de archivo, como un error de lógica de aplicación o un error de disco.
- Compruebe la clave de estado del archivo después de cada solicitud de entrada o salida.

  Si la clave de estado de archivo contiene un valor distinto de 0, el programa puede emitir un mensaje de error o puede realizar una acción basada en ese valor.

No tiene que restablecer el código de clave de estado de archivo, porque se establece después de cada intento de entrada o salida.

Además de la clave de estado de archivo, puede codificar un segundo identificador en la cláusula FILE STATUS para obtener información más detallada sobre las solicitudes de entrada o salida del sistema de archivos. Para obtener detalles, consulte la tarea relacionada sobre los códigos de estado del sistema de archivos.

Puede utilizar la clave de estado de archivo sola o junto con la frase INVALID KEY , o para complementar la declarativa EXCEPTION o ERROR . El uso de la clave de estado de archivo de esta forma le proporciona información precisa sobre los resultados de cada operación de entrada o salida.

"Ejemplo: clave de estado de archivo" en la página 182
"Ejemplo: comprobar los códigos de estado del sistema de archivos" en la página 183

Tareas relacionadas
"Configuración de un campo para el estado de archivo" en la página 140
"Utilización de códigos de estado del sistema de archivos" en la página 182
"Codificación de frases INVALID KEY" en la página 184
"Búsqueda y manejo de errores de entrada-salida" en la página 320

Referencias relacionadas
cláusula FILE STATUS (*COBOL for Linux en x86 Consulta de lenguaje*)
Clave de estado de archivo (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: clave de estado de archivo

El ejemplo siguiente muestra cómo puede realizar una comprobación simple de la clave de estado de archivo después de abrir un archivo.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  SIMCHK.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT MAINFILE ASSIGN TO AS-MAINA
    FILE STATUS IS MAINFILE-CHECK-KEY
    . . .
DATA DIVISION.
. . .
WORKING-STORAGE SECTION.
01  MAINFILE-CHECK-KEY       PIC X(2).
. . .
PROCEDURE DIVISION.
    OPEN INPUT MAINFILE
    IF MAINFILE-CHECK-KEY NOT = "00"
        DISPLAY "Nonzero file status returned from OPEN " MAINFILE-CHECK-KEY
    . . .
```

Utilización de códigos de estado del sistema de archivos

A menudo, el estado del archivo de dos dígitos clave es demasiado general para identificar el resultado de una solicitud de entrada o salida. Puede obtener información más detallada sobre las solicitudes de sistema de archivos Db2, LSQ, QSAM, RSD, SdU, SFS y STL codificando un segundo elemento de datos en la cláusula FILE STATUS .

Acerca de esta tarea

```
FILE STATUS IS data-name-1 data-name-8
```

En el ejemplo anterior, el elemento de datos *data-name-1* especifica la clave de estado de archivo COBOL de dos dígitos , que debe ser un elemento nacional de categoría alfanumérica o de categoría de dos caracteres, o un elemento decimal con zona de dos dígitos o un elemento decimal nacional. El elemento

de datos *data-name-8* especifica un elemento de datos que contiene el código de estado del sistema de archivos si la clave de estado de archivo COBOL no es cero. *data-name-8* tiene al menos 6 bytes de longitud, y debe ser un elemento alfanumérico.

Archivos LSQ, QSAM, RSD, SFS, STL y SdU : Para solicitudes de entrada y salida del sistema de archivos LSQ, QSAM, RSD, SFS, STL y SdU , si *data-name-8* tiene una longitud de 6 bytes, contiene el código de estado de archivo. Si *data-name-8* tiene más de 6 bytes, también contiene un mensaje con más información:

```
01  my-file-status-2.
    02 exception-return-value PIC 9(6).
    02 additional-info        PIC X(100).
```

`exception-return-value` contiene un valor que puede refinar más el error anotado en `FILE STATUS`. El `additional-info` contiene información de diagnóstico adicional del error anotado en `exception-return-value` para ayudarle a diagnosticar el problema.

Archivos deDb2 : Para solicitudes de entrada y salida del sistema de archivos de Db2 , definir *data-name-8* como un elemento de grupo. Por ejemplo:

```
01  FileStatus2.
    02 FS2-SQLCODE  PICTURE S9(9) COMP.
    02 FS2-SQLSTATE PICTURE X(5).
```

Los valores de tiempo de ejecución en `FS2-SQLCODE` y `FS2-SQLSTATE` representan información de comentarios de SQL para la operación completada anteriormente.

“Ejemplo: comprobar los códigos de estado del sistema de archivos” en la página 183

Tareas relacionadas
"Arreglo de diferencias causadas por elementos de lenguaje" en la *Guía de migración*

Referencias relacionadas
“Sistema de archivos Db2” en la página 123
“Sistema de archivos QSAM” en la página 124
“Sistema de archivos MongoDB” en la página 125
“Sistema de archivos SFS” en la página 126
“Sistema de archivos STL” en la página 126
cláusula FILE STATUS (*COBOL for Linux en x86 Consulta de lenguaje*)
Clave de estado de archivo (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: comprobar los códigos de estado del sistema de archivos

El ejemplo siguiente lee un archivo indexado a partir del quinto registro y comprueba la clave de estado de archivo después de cada solicitud de entrada o salida. Los códigos de estado de archivo se visualizan si la clave de estado de archivo no es cero.

Este ejemplo también ilustra el aspecto que puede tener la salida de este programa si el archivo que se está procesando contiene seis registros.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. EXAMPLE.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT FILESYSFILE ASSIGN TO FILESYSFILE
    ORGANIZATION IS INDEXED
    ACCESS DYNAMIC
    RECORD KEY IS FILESYSFILE-KEY
    FILE STATUS IS FS-CODE, FILESYS-CODE.
DATA DIVISION.
FILE SECTION.
FD  FILESYSFILE
    RECORD  30.
```

```
01  FILESYSFILE-REC.
    10 FILESYSFILE-KEY          PIC X(6).
    10 FILLER                   PIC X(24).
WORKING-STORAGE SECTION.
01  RETURN-STATUS.
    05 FS-CODE                  PIC XX.
    05 FILESYS-CODE             PIC X(6).
PROCEDURE DIVISION.
    OPEN  INPUT FILESYSFILE.
    DISPLAY "OPEN INPUT FILESYSFILE FS-CODE: " FS-CODE.

    IF FS-CODE NOT = "00"
       PERFORM FILESYS-CODE-DISPLAY
       STOP RUN
    END-IF.

    MOVE "000005" TO FILESYSFILE-KEY.
    START FILESYSFILE KEY IS EQUAL TO FILESYSFILE-KEY.
    DISPLAY "START FILESYSFILE KEY="  FILESYSFILE-KEY
            " FS-CODE: "  FS-CODE.

    IF FS-CODE NOT = "00"
       PERFORM FILESYS-CODE-DISPLAY
    END-IF.

    IF FS-CODE = "00"
       PERFORM READ-NEXT UNTIL FS-CODE NOT = "00"
    END-IF.

    CLOSE FILESYSFILE.
    STOP RUN.

READ-NEXT.
    READ FILESYSFILE NEXT.
    DISPLAY "READ NEXT FILESYSFILE FS-CODE: " FS-CODE.
    IF FS-CODE NOT = "00"
       PERFORM FILESYS-CODE-DISPLAY
    END-IF.
    DISPLAY FILESYSFILE-REC.

FILESYS-CODE-DISPLAY.
    DISPLAY "FILESYS-CODE ==>", FILESYS-CODE.
```

Codificación de frases INVALID KEY

Puede incluir una frase INVALID KEY en las sentencias READ, START, WRITE, REWRITEy DELETE para los archivos indexados y relativos de . A la frase INVALID KEY se le otorga el control si se produce un error de entrada o salida debido a una clave de índice defectuosa.

Acerca de esta tarea

Utilice la cláusula FILE STATUS con la frase INVALID KEY para evaluar la clave de estado y determinar la condición INVALID KEY específica.

Las frases INVALID KEY difieren de los declarativos ERROR de varias maneras. Frases de INVALID KEY :

- Operar sólo para tipos limitados de errores. Las declarativas de ERROR abarcan todos los formularios.
- Se codifican directamente con la sentencia de entrada o salida. Las declarativas de ERROR se codifican por separado.
- Son específicos para una sola operación de entrada o salida. Las declarativas de ERROR son más generales.

Si codifica INVALID KEY en una sentencia que provoca una condición INVALID KEY , el control se transfiere a la sentencia imperativa INVALID KEY . Los declarativos de ERROR que ha codificado no se realizan.

Si codifica una frase NOT INVALID KEY , sólo se realiza si la sentencia se completa correctamente. Si la operación falla debido a una condición distinta de INVALID KEY, no se realiza la frase INVALID KEY ni la frase NOT INVALID KEY . En su lugar, después de que el programa realice cualquier declaración ERROR asociada, el control pasa al final de la sentencia.

“Ejemplo: FILE STATUS y INVALID KEY” en la página 185

Ejemplo: FILE STATUS y INVALID KEY

El ejemplo siguiente muestra cómo puede utilizar el código de estado de archivo y la frase INVALID KEY para determinar más específicamente por qué ha fallado una sentencia de entrada o salida.

Supongamos que tiene un archivo que contiene registros de cliente principales y que necesita actualizar algunos de estos registros con información de un archivo de actualización de transacciones. El programa lee cada registro de transacción, busca el registro correspondiente en el archivo principal y realiza las actualizaciones necesarias. Los registros de ambos archivos contienen un campo para un número de cliente y cada registro del archivo principal tiene un número de cliente exclusivo.

La entrada FILE-CONTROL para el archivo principal de registros de cliente incluye sentencias que definen la organización indexada, el acceso aleatorio, MAIN-CUSTOMER-NUMBER como clave de registro principal y CUSTOMER-FILE-STATUS como clave de estado de archivo.

```
.
.  (read the update transaction record)
.
MOVE "TRUE" TO TRANSACTION-MATCH
MOVE UPDATE-CUSTOMER-NUMBER TO MAIN-CUSTOMER-NUMBER
READ MAIN-CUSTOMER-FILE INTO WS-CUSTOMER-RECORD
  INVALID KEY
    DISPLAY "MAIN CUSTOMER RECORD NOT FOUND"
    DISPLAY "FILE STATUS CODE IS: " CUSTOMER-FILE-STATUS
    MOVE "FALSE" TO TRANSACTION-MATCH
END-READ
```

Manejo de errores al llamar a programas

Cuando un programa llama dinámicamente a un programa compilado por separado, es posible que el programa llamado no esté disponible. Por ejemplo, es posible que el sistema se haya quedado sin almacenamiento o que no pueda localizar el objeto de programa . Si la sentencia CALL no tiene una frase ON EXCEPTION o ON OVERFLOW , es posible que la aplicación termine de forma anómala.

Acerca de esta tarea

Utilice la frase ON EXCEPTION para realizar una serie de sentencias y para realizar su propio manejo de errores. Por ejemplo, en el fragmento de código siguiente, si el programa REPORTA no está disponible, el control pasa a la frase ON EXCEPTION .

```
MOVE "REPORTA" TO REPORT-PROG
CALL REPORT-PROG
  ON EXCEPTION
    DISPLAY "Program REPORTA not available, using REPORTB."
    MOVE "REPORTB" TO REPORT-PROG
    CALL REPORT-PROG
    END-CALL
END-CALL
```

La frase ON EXCEPTION sólo se aplica a la disponibilidad del programa llamado en su carga inicial. Si el programa llamado se carga pero falla por cualquier otro motivo (como la inicialización), no se realiza la frase ON EXCEPTION .

Parte 2. Habilitación de programas para entornos internacionales

Capítulo 10. Tratamiento de datos en un entorno internacional

COBOL para Linux soporta Unicode UTF-16 como datos de caracteres nacionales en tiempo de ejecución. UTF-16 es una codificación Unicode de anchura fija que proporciona una forma coherente y eficaz de codificar texto sin formato. Utilizando UTF-16, puede desarrollar software que funcionará con varios idiomas nacionales.

Acerca de esta tarea

Utilice estos recursos COBOL para codificar y compilar programas que procesan datos nacionales y órdenes de clasificación culturalmente sensibles para dichos datos:

- Tipos de datos y literales:
  - Tipos de datos de tipo carácter, definidos con la cláusula `USAGE NATIONAL` y una cláusula `PICTURE` que define datos de categoría nacional, nacional editada o numérica editada
  - Tipos de datos numéricos, definidos con la cláusula `USAGE NATIONAL` y una cláusula `PICTURE` que define un elemento de datos numérico (un *elemento decimal nacional*) o un elemento de datos de coma flotante externo (un *elemento de coma flotante nacional*)
  - Literales nacionales, especificados con el prefijo literal `N` o `NX`
  - Constante figurativa `ALL` *literal-nacional*
  - Constantes figurativas `QUOTE`, `SPACE`, `HIGH-VALUE`, `LOW-VALUE`o `ZERO`, que tienen valores de caracteres nacionales (UTF-16) cuando se utilizan en contextos de caracteres nacionales
- Las sentencias COBOL que se muestran en la referencia relacionada siguiente sobre sentencias COBOL y datos nacionales
- Funciones intrínsecas:
  - `NATIONAL-OF` para convertir una serie de caracteres alfanuméricos o de doble byte (DBCS) en `USAGE NATIONAL` (UTF-16)
  - `DISPLAY-OF` para convertir una serie de caracteres nacionales a `USAGE DISPLAY` en una página de códigos seleccionada (EBCDIC, ASCII, EUC, o UTF-8)
  - Las otras funciones intrínsecas que se muestran en la referencia relacionada a continuación sobre las funciones intrínsecas y los datos nacionales
- La cláusula `GROUP-USAGE NATIONAL` para definir grupos que contienen sólo elementos de datos `USAGE NATIONAL` y que se comportan como elementos nacionales de categoría elemental en la mayoría de las operaciones
- Opciones de compilador:
  - `NSYMBOL` para controlar si se utiliza el proceso nacional o DBCS para el símbolo `N` en literales y cláusulas `PICTURE`
  - `NCOLLSEQ` para especificar el orden de clasificación para la comparación de operandos nacionales

También puede aprovechar las conversiones implícitas de elementos de datos alfanuméricos o DBCS a la representación nacional. El compilador realiza estas conversiones (en la mayoría de los casos) cuando mueve estos elementos a elementos de datos nacionales, o compara estos elementos con elementos de datos nacionales.

Conceptos relacionados
“Unicode y la codificación de caracteres de idioma” en la página 190
“Grupos nacionales” en la página 198

Tareas relacionadas
“Utilización de datos nacionales (Unicode) en COBOL” en la página 191
“Conversión a o desde representación nacional (Unicode)” en la página 199
“Proceso de datos UTF-8 utilizando tipos de datos UTF-16 (nacional)” en la página 208
“Procesando datos en chino GB 18030” en la página 209
“Comparación de datos nacionales (UTF-16)” en la página 205
“Codificación para el uso del soporte DBCS” en la página 210
Capítulo 11, “Establecimiento del entorno local”, en la página 213

Referencias relacionadas
“Sentencias COBOL y datos nacionales” en la página 193
“Funciones intrínsecas y datos nacionales” en la página 196
“NCOLLSEQ” en la página 290
“SÍMBOLO” en la página 290
Clases y categorías de datos (*COBOL for Linux en x86 Consulta de lenguaje*)
Categorías de datos y reglas PICTURE
(*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia MOVE (*COBOL for Linux en x86 Consulta de lenguaje*)
Condiciones de relación generales (*COBOL for Linux en x86 Consulta de lenguaje*)

Unicode y la codificación de caracteres de idioma

COBOL para Linux proporciona soporte de tiempo de ejecución básico para Unicode, que puede manejar decenas de miles de caracteres que cubren todos los caracteres y símbolos utilizados habitualmente en el mundo.

Un *juego de caracteres* es un conjunto definido de caracteres, pero no está asociado con una representación codificada. Un *juego de caracteres codificado* (al que también se hace referencia en esta documentación como una *página de códigos*) es un conjunto de reglas inequívocas que relacionan los caracteres del conjunto con su representación codificada.Cada página de códigos tiene un nombre y es como una tabla que configura los símbolos para representar un juego de caracteres; cada símbolo está asociado con un patrón de bits exclusivo, o *elemento de código*.Cada página de códigos también tiene un *identificador de juego de caracteres codificados* (*CCSID*), que es un valor de 1 a 65.536.

Unicode tiene varios esquemas de codificación, denominados *Unicode Transformation Format (UTF)*, como por ejemplo UTF-8, UTF-16y UTF-32. COBOL para Linux utiliza UTF-16 (CCSID 1200) en formato little-endian como representación de literales nacionales y elementos de datos que tienen USAGE `NATIONAL`.

UTF-8 representa los caracteres invariables ASCII a-z, A-Z, 0-9, y determinados caracteres especiales como ' @,. +-=/* () de la misma forma que se representan en ASCII. UTF-16 representa estos caracteres como `NX'`*nn*`00'`, donde `X'`*nn*`'` es la representación del carácter en ASCII.

Por ejemplo, la serie `'ABC'` se representa en UTF-16 como `NX'410042004300'`. En UTF-8, `'ABC'` se representa como `X'414243'`.

Se utilizan una o más *unidades de codificación* para representar un carácter de un juego de caracteres codificado. Para UTF-16, una unidad de codificación toma 2 bytes de almacenamiento. Cualquier carácter definido en cualquier página de códigos EBCDIC, ASCII o EUC se representa en una unidad de codificación UTF-16 cuando el carácter se convierte a la representación de datos nacional.

Consideraciones sobre varias plataformas: Enterprise COBOL for z/OS y COBOL for AIX support UTF-16 en formato big-endian en datos nacionales. De forma predeterminada, COBOL para Linux da soporte a UTF-16 en formato little-endian en los datos nacionales. Si está portando datos Unicode codificados en la representación UTF-16BE a COBOL para Linux desde otra plataforma, debe convertir esos datos a UTF-16 en formato little-endian para procesar los datos como datos nacionales, o utilice la opción de compilador UTF16 para cambiar la forma en que el compilador trata el endianness UTF-16 . Con COBOL para Linux, puede realizar estas conversiones utilizando la función intrínseca NATIONAL-OF.

Tareas relacionadas
“Conversión a o desde representación nacional (Unicode)” en la página 199

Referencias relacionadas
“Almacenamiento de datos de caracteres” en la página 205
“Entornos locales y páginas de códigos soportadas” en la página 217
Juegos de caracteres y páginas de códigos (*COBOL for Linux en x86 Consulta de lenguaje*)

Utilización de datos nacionales (Unicode) en COBOL

En COBOL para Linux, puede especificar datos nacionales (UTF-16) de varias maneras.

Acerca de esta tarea

Estos tipos de datos nacionales están disponibles:

- Elementos de datos nacionales (categorías nacionales, nacionales editadas y numéricas editadas)
- Literales nacionales
- Constantes figurativas como caracteres nacionales
- Elementos de datos numéricos (decimal nacional y coma flotante nacional)

Además, puede definir grupos nacionales que contengan sólo elementos de datos que tengan explícita o implícitamente `USAGE NATIONAL`, y que se comporten de la misma forma que los elementos de datos nacionales de categoría elemental en la mayoría de las operaciones.

Estas declaraciones afectan a la cantidad de almacenamiento que se necesita.

Conceptos relacionados
“Unicode y la codificación de caracteres de idioma” en la página 190
“Grupos nacionales” en la página 198

Tareas relacionadas
“Definición de elementos de datos nacionales” en la página 191
“Utilización de literales nacionales” en la página 192
“Utilización de constantes figurativas de carácter nacional” en la página 196
“Definición de elementos de datos numéricos nacionales” en la página 197
“Utilización de grupos nacionales” en la página 202
“Conversión a o desde representación nacional (Unicode)” en la página 199
“Comparación de datos nacionales (UTF-16)” en la página 205

Referencias relacionadas
“Almacenamiento de datos de caracteres” en la página 205
Clases y categorías de datos (*COBOL for Linux en x86 Consulta de lenguaje*)

Definición de elementos de datos nacionales

Defina elementos de datos nacionales con la cláusula `USAGE NATIONAL` para que contengan series de caracteres nacionales (UTF-16) .

Acerca de esta tarea

Puede definir elementos de datos nacionales de las categorías siguientes:

- Nacional
- Nacional-editado
- Numérico-editado

Para definir un elemento de datos nacional de categoría , codifique una cláusula `PICTURE` que contenga sólo uno o más `PICTURE` símbolos `N`.

Para definir un elemento de datos editado a nivel nacional, codifique una cláusula `PICTURE` que contenga al menos uno de los símbolos siguientes:

- Símbolo N
- Símbolo de edición de inserción simple B, 0o /

Para definir un elemento de datos editado numérico de class national, codifique una cláusula PICTURE que defina un elemento editado numérico (por ejemplo, -$999.99) y codifique una cláusula USAGE NATIONAL . Puede utilizar un elemento de datos editado de forma numérica que tenga USAGE NATIONAL de la misma forma que utiliza un elemento editado de forma numérica que tenga USAGE DISPLAY.

También puede definir un elemento de datos como numérico editado codificando la cláusula BLANK WHEN ZERO para un elemento elemental definido como numérico por su cláusula PICTURE .

Si codifica una cláusula PICTURE pero no codifica una cláusula USAGE para elementos de datos que contienen sólo uno o más PICTURE símbolos N, puede utilizar la opción de compilador NSYMBOL(NATIONAL) para asegurarse de que dichos elementos se traten como elementos de datos nacionales en lugar de como elementos DBCS.

Tareas relacionadas
“Visualización de datos numéricos” en la página 37

Referencias relacionadas
“SÍMBOLO” en la página 290
cláusula BLANK WHEN ZERO (*COBOL for Linux en x86 Consulta de lenguaje*)

Utilización de literales nacionales

Para especificar literales nacionales, utilice el carácter de prefijo N y compile con la opción NSYMBOL(NATIONAL).

Acerca de esta tarea

Puede utilizar cualquiera de estas notaciones:

- N"*character-data*"
- N'*character-data*'

Si compila con la opción NSYMBOL(DBCS), el carácter de prefijo literal N especifica un literal DBCS, no un literal nacional.

Para especificar un literal nacional como valor hexadecimal, utilice el prefijo NX. Puede utilizar cualquiera de estas notaciones:

- NX"*hexadecimal-digits*"
- NX'*hexadecimal-digits*'

Cada una de las sentencias MOVE siguientes establece el elemento de datos nacional Y en el valor UTF-16 de los caracteres 'AB':

```
01 Y pic NN usage national.
. . .
    Move NX"41004200" to Y
    Move N"AB"        to Y
    Move "AB"         to Y
```

No utilice literales hexadecimales alfanuméricos en contextos que llamen a literales nacionales, porque tal uso es fácilmente incomprendido. Por ejemplo, la sentencia siguiente también hace que se muevan los caracteres UTF-16 'AB' (no el patrón de bits hexadecimal 4142) a Y, donde Y se define como USAGE NATIONAL:

```
Move X"4142" to Y
```

No puede utilizar literales nacionales en el SPECIAL-NAMES párrafo o como nombres de programa. Puede utilizar un literal nacional para nombrar un método orientado a objetos en el párrafo METHOD-ID o para especificar un nombre de método en una sentencia INVOKE .

Utilice la opción de compilador SOSI para controlar cómo se manejan los caracteres de desplazamiento a teclado ideográfico dentro de un literal nacional.

Tareas relacionadas
"Utilización de literales" en la página 21

Referencias relacionadas
"SÍMBOLO" en la página 290
"SOSI" en la página 295
Literales nacionales (*COBOL for Linux en x86 Consulta de lenguaje*)

Sentencias COBOL y datos nacionales

Puede utilizar datos nacionales con PROCEDURE DIVISION y las sentencias de dirección de compilador que se muestran en la tabla siguiente.

Tabla 16. ***Sentencias COBOL y datos nacionales***

sentencia COBOL	Puede ser nacional	Comentario	Para obtener más información
ACCEPT	*identifier-1*, *identifier-2*	*identifier-1* es convertido desde página de códigos indicada por la entorno local de ejecución sólo si la entrada procede del terminal.	"Asignación de entrada desde una pantalla o archivo (ACCEPT)" en la página 31
ADD	Todos los identificadores pueden ser elementos numéricos que tienen USAGE NATIONAL. *identifier-3* (GIVING) puede ser numérico editado con USAGE NATIONAL.		"Utilización de COMPUTE y otras sentencias aritméticas" en la página 49
CALL	*identifier-2*, *identifier-3*, *identifier-4*, *identifier-5*; *literal-2*, *literal-3*		"Pasar datos" en la página 473
COMPUTE	*identifier-1* puede ser numérico o numérico editado con USAGE NATIONAL. *expresión-aritmética* puede contener elementos numéricos que tengan USAGE NATIONAL.		"Utilización de COMPUTE y otras sentencias aritméticas" en la página 49
COPY . . . REPLACING	*operand-1*, *operand-2* de la frase REPLACING		Capítulo 14, "Sentencias de direccionamiento de compilador", en la página 309

Tabla 16. ***Sentencias COBOL y datos nacionales*** *(continuación)*			
sentencia COBOL	**Puede ser nacional**	**Comentario**	**Para obtener más información**
DISPLAY	*identifier-1*	*identifier-1* se convertido a la página de códigos asociada con el entorno local actual.	“Visualización de valores en una pantalla o en un archivo (DISPLAY)” en la página 32
DIVIDE	Todos los identificadores pueden ser elementos numéricos que tienen USAGE NATIONAL. *identifier-3* (GIVING) y *identifier-4* (REMAINDER) pueden ser numéricos editados con USAGE NATIONAL.		“Utilización de COMPUTE y otras sentencias aritméticas” en la página 49
INITIALIZE	*identifier-1*; *identifier-2* o *literal-1* de la frase REPLACING	Si especifica REPLACING NATIONAL o REPLACING NATIONAL-EDITED, *identifier-2* o *literal-1* debe ser válido como operando de envío en un movimiento a *identifier-1*.	“Ejemplos: inicialización de elementos de datos” en la página 24
INSPECT	Todos los identificadores y literales. (*identifier-2*, el elemento de datos entero TALLYING , puede tener USAGE NATIONAL.)	Si alguno de ellos (que no sea *identifier-2*, el identificador TALLYING) tiene USAGE NATIONAL, todos deben ser nacionales.	“Recuento y sustitución de elementos de datos (INSPECT)” en la página 104
MERGE	Fusionar claves, si especifica NCOLLSEQ(BIN)	La frase COLLATING SEQUENCE no se aplica.	“Establecimiento de criterios de ordenación o fusión” en la página 169
MOVE	Tanto el remitente como el destinatario, o solo el destinatario	Las conversiones implícitas se realizan para operandos MOVE válidos.	“Asignación de valores a elementos de datos elementales (MOVE)” en la página 28 “Asignación de valores a elementos de datos de grupo (MOVE)” en la página 29
MULTIPLY	Todos los identificadores pueden ser elementos numéricos que tienen USAGE NATIONAL. *identifier-3* (GIVING) puede ser numérico editado con USAGE NATIONAL.		“Utilización de COMPUTE y otras sentencias aritméticas” en la página 49

Tabla 16. ***Sentencias COBOL y datos nacionales*** *(continuación)*			
sentencia COBOL	**Puede ser nacional**	**Comentario**	**Para obtener más información**
SEARCH ALL (búsqueda binaria)	Tanto el elemento de datos clave como su objeto de comparación	El elemento de datos clave y su objeto de comparación deben ser compatibles de acuerdo con las reglas de comparación. Si el objeto de comparación es de clase nacional, la clave también debe ser.	"Realización de una búsqueda binaria (SEARCH ALL)" en la página 79
SORT	Claves de ordenación, si especifica NCOLLSEQ(BIN)	La frase COLLATING SEQUENCE no se aplica.	"Establecimiento de criterios de ordenación o fusión" en la página 169
STRING	Todos los identificadores y literales. (*identifier-4*, el elemento de datos entero POINTER , puede tener USAGE NATIONAL.)	Si *identifier-3*, el elemento de datos receptor, es nacional, todos los identificadores y literales (que no sean *identifier-4*, el identificador POINTER) deben ser nacionales.	"Unión de elementos de datos (STRING)" en la página 95
SUBTRACT	Todos los identificadores pueden ser elementos numéricos que tienen USAGE NATIONAL. *identifier-3* (GIVING) puede ser numérico editado con USAGE NATIONAL.		"Utilización de COMPUTE y otras sentencias aritméticas" en la página 49
UNSTRING	Todos los identificadores y literales. (*identifier-6* y *identifier-7*, los elementos de datos enteros COUNT y TALLYING , respectivamente, puede tener USAGE NATIONAL.)	Si *identifier-4*, un elemento de datos de recepción, tiene USAGE NATIONAL, el elemento de datos de envío y cada delimitador deben tener USAGE NATIONAL, y cada literal debe ser nacional.	"División de elementos de datos (UNSTRING)" en la página 97
XML GENERATE	*identifier-1* (el documento XML generado); *identifier-2* (el campo o campos de origen); *identifier-4* o *literal-4* (el identificador de espacio de nombres); *identifier-5* o *literal-5* (el prefijo de espacio de nombres)		Capítulo 20, "Producción de salida XML", en la página 439
XML PARSE	*identifier-1* (el documento XML)	El registro especial XML-NTEXT contiene fragmentos de documento de caracteres nacionales durante el análisis.	Capítulo 19, "Procesando entrada XML", en la página 419

Tareas relacionadas
“Definición de datos numéricos” en la página 35
“Visualización de datos numéricos” en la página 37
“Utilización de datos nacionales (Unicode) en COBOL” en la página 191
“Comparación de datos nacionales (UTF-16)” en la página 205

Referencias relacionadas
“NCOLLSEQ” en la página 290
Clases y categorías de datos (*COBOL for Linux en x86 Consulta de lenguaje*)

Funciones intrínsecas y datos nacionales

Puede utilizar argumentos de clase nacional con las funciones intrínsecas que se muestran en la tabla siguiente.

Tabla 17. ***Funciones intrínsecas y datos de caracteres nacionales***

Función intrínseca	**Tipo de función**	**Para obtener más información**
DISPLAY-OF	Alfanumérico	“Conversión de nacional a alfanumérico (DISPLAY-OF)” en la página 200
LENGTH	Entero	“Búsqueda de la longitud de los elementos de datos” en la página 112
LOWER-CASE, UPPER-CASE	Nacional	“Cambio de mayúsculas/minúsculas (UPPER-CASE, LOWER-CASE)” en la página 106
NUMVAL, NUMVAL-C,	Numérico	“Conversión a números (NUMVAL, NUMVAL-C)” en la página 107
MAX, MIN	Nacional	“Búsqueda del elemento de datos más grande o más pequeño” en la página 109
ORD-MAX, ORD-MIN	Entero	“Búsqueda del elemento de datos más grande o más pequeño” en la página 109
REVERSE	Alfanumérico o nacional	“Transformación a orden inverso (REVERSE)” en la página 107

Puede utilizar argumentos decimales nacionales siempre que se permitan los argumentos decimales con zona. Puede utilizar argumentos de coma flotante nacionales siempre que se permitan los argumentos de coma flotante de visualización. (Consulte la referencia relacionada a continuación sobre los argumentos para obtener una lista completa de las funciones intrínsecas que pueden tomar argumentos enteros o numéricos.)

Tareas relacionadas
“Definición de datos numéricos” en la página 35
“Utilización de datos nacionales (Unicode) en COBOL” en la página 191

Referencias relacionadas
Argumentos (*COBOL for Linux en x86 Consulta de lenguaje*)
Clases y categorías de datos (*COBOL for Linux en x86 Consulta de lenguaje*)
Funciones intrínsecas (*COBOL for Linux en x86 Consulta de lenguaje*)

Utilización de constantes figurativas de carácter nacional

Puede utilizar la constante figurativa ALL *literal-nacional* en un contexto que requiera caracteres nacionales. ALL *literal-nacional* representa toda o parte de la serie generada por concatenaciones sucesivas de las unidades de codificación que componen el literal nacional.

Acerca de esta tarea

Puede utilizar las constantes figurativas QUOTE, SPACE, HIGH-VALUE, LOW-VALUE, o ZERO en un contexto que requiera caracteres nacionales, como, por ejemplo, una sentencia MOVE , un movimiento implícito o una condición de relación que tiene operandos nacionales. En estos contextos, la constante figurativa representa un valor de carácter nacional (UTF-16).

Cuando se utiliza la constante figurativa HIGH-VALUE en un contexto que requiere caracteres nacionales, su valor es NX'FFFF'. Cuando se utiliza LOW-VALUE en un contexto que requiere caracteres nacionales, su valor es NX'0000'. Puede utilizar HIGH-VALUE o LOW-VALUE en un contexto que requiera caracteres nacionales sólo si la opción de compilador NCOLLSEQ(BIN) está en vigor.

Restricciones: No debe utilizar HIGH-VALUE o el valor asignado desde HIGH-VALUE de una forma que dé como resultado la conversión del valor de una representación de datos a otra (por ejemplo, entre USAGE DISPLAY y USAGE NATIONAL, o entre ASCII y EBCDIC cuando la opción de compilador CHAR(EBCDIC) está en vigor). X'FF' (el valor de HIGH-VALUE en un contexto alfanumérico cuando se utiliza la secuencia de clasificación EBCDIC) no representa un carácter EBCDIC o ASCII válido y NX'FFFF' no representa un carácter nacional válido.La conversión de un valor de este tipo a otra representación hace que se utilice un *carácter de sustitución* (no X'FF' ni NX'FFFF'). Considere el ejemplo siguiente:

```
01 natl-data  PIC NN  Usage National.
01 alph-data  PIC XX.
. . .
    MOVE HIGH-VALUE TO natl-data, alph-data
    IF natl-data = alph-data. . .
```

La sentencia IF anterior se evalúa como falsa aunque cada uno de sus operandos se haya establecido en HIGH-VALUE. Antes de que un operando alfanumérico elemental se compare con un operando nacional, el operando alfanumérico se trata como si se trasladara a un elemento de datos nacional temporal, y los caracteres alfanuméricos se convierten a los caracteres nacionales correspondientes. Sin embargo, cuando X'FF' se convierte a UTF-16, el elemento UTF-16 obtiene un valor de carácter de sustitución y, por lo tanto, no se compara igualmente con NX'FFFF'.

Tareas relacionadas
“Conversión a o desde representación nacional (Unicode)” en la página 199
“Comparación de datos nacionales (UTF-16)” en la página 205

Referencias relacionadas
“char” en la página 272
“NCOLLSEQ” en la página 290
Constantes figurativas (*COBOL for Linux en x86 Consulta de lenguaje*)
DISPLAY-OF (*COBOL for Linux en x86 Consulta de lenguaje*)

Definición de elementos de datos numéricos nacionales

Defina elementos de datos con la cláusula USAGE NATIONAL para contener datos numéricos representados en caracteres nacionales (UTF-16). Puede definir elementos decimales nacionales y elementos de coma flotante nacionales.

Acerca de esta tarea

Para definir un elemento decimal nacional, codifique una cláusula PICTURE que contenga sólo los símbolos 9, P, Sy V. Si la cláusula PICTURE contiene S, la cláusula SIGN IS SEPARATE debe estar en vigor para ese elemento.

Para definir un elemento de coma flotante nacional, codifique una cláusula PICTURE que defina un elemento de coma flotante (por ejemplo, +99999.9E-99).

Puede utilizar elementos decimales nacionales de la misma forma que utiliza elementos decimales con zona. Puede utilizar elementos de coma flotante nacionales de la misma forma que utiliza elementos de coma flotante de visualización.

Tareas relacionadas
"Definición de datos numéricos" en la página 35
"Visualización de datos numéricos" en la página 37

Referencias relacionadas
Cláusula SIGN (*COBOL for Linux en x86 Consulta de lenguaje*)

Grupos nacionales

Los grupos nacionales, que se especifican explícita o implícitamente con la cláusula GROUP-USAGE NATIONAL , sólo contienen elementos de datos que tienen USAGE NATIONAL. En la mayoría de los casos, un elemento de grupo nacional se procesa como si se hubiera redefinido como un elemento nacional de categoría elemental descrito como PIC N(*m*), donde *m* es el número de caracteres nacionales (UTF-16) del grupo.

Sin embargo, para algunas operaciones en grupos nacionales (al igual que para algunas operaciones en grupos alfanuméricos), se aplica la semántica de grupo. Estas operaciones (por ejemplo, MOVE CORRESPONDING y INITIALIZE) reconocen o procesan los elementos elementales dentro del grupo nacional.

Cuando sea posible, utilice grupos nacionales en lugar de grupos alfanuméricos que contengan elementos USAGE NATIONAL . Los grupos nacionales ofrecen varias ventajas para el tratamiento de datos nacionales en comparación con el tratamiento de datos nacionales dentro de grupos alfanuméricos:

- Cuando mueve un grupo nacional a un elemento de datos más largo que tiene USAGE NATIONAL, el elemento receptor se rellena con caracteres nacionales. Por el contrario, si mueve un grupo alfanumérico que contiene caracteres nacionales a un grupo alfanumérico más largo que contiene caracteres nacionales, se utilizan espacios alfanuméricos para el relleno. Como resultado, podría producirse un mal manejo de los elementos de datos.
- Cuando mueve un grupo nacional a un elemento de datos más corto que tiene USAGE NATIONAL, el grupo nacional se trunca en los límites de caracteres nacionales. Por el contrario, si mueve un grupo alfanumérico que contiene caracteres nacionales a un grupo alfanumérico más corto que contiene caracteres nacionales, puede producirse un truncamiento entre los 2 bytes de un carácter nacional.
- Cuando se mueve un grupo nacional a un elemento editado a nivel nacional o numérico, se edita el contenido del grupo. Por el contrario, si mueve un grupo alfanumérico a un elemento editado, no se realiza ninguna edición.
- Cuando se utiliza un grupo nacional como operando en una sentencia STRING, UNSTRINGo INSPECT :
  - El contenido del grupo se procesa como caracteres nacionales en lugar de como caracteres de un solo byte.
  - Los operandos TALLYING y POINTER funcionan a nivel lógico de caracteres nacionales.
  - El operando de grupo nacional se soporta con una mezcla de otros tipos de operando nacionales.

  Por el contrario, si utiliza un grupo alfanumérico que contiene caracteres nacionales en estos contextos, los caracteres se procesan byte a byte. Como resultado, podría producirse un manejo no válido o una corrupción de datos.

Grupos de **USAGE NATIONAL :** un elemento de grupo puede especificar la cláusula USAGE NATIONAL a nivel de grupo como una abreviatura conveniente para el USAGE de cada uno de los elementos de datos elementales del grupo. Sin embargo, un grupo de este tipo *no* es un grupo nacional, sino un grupo alfanumérico, y se comporta en muchas operaciones, como movimientos y comparaciones, como un elemento de datos elemental de USAGE DISPLAY (excepto que no se produce ninguna edición o conversión de datos).

Tareas relacionadas
“Asignación de valores a elementos de datos de grupo (MOVE)” en la página 29
“Unión de elementos de datos (STRING)” en la página 95
“División de elementos de datos (UNSTRING)” en la página 97
“Recuento y sustitución de elementos de datos (INSPECT)” en la página 104
“Utilización de grupos nacionales” en la página 202

Referencias relacionadas
cláusula GROUP-USAGE (*COBOL for Linux en x86 Consulta de lenguaje*)

Conversión a o desde representación nacional (Unicode)

Puede convertir implícita o explícitamente elementos de datos a representación nacional (UTF-16).

Acerca de esta tarea

Puede convertir implícitamente datos alfabéticos, alfanuméricos, DBCS o enteros a datos nacionales utilizando la sentencia MOVE . Las conversiones implícitas también tienen lugar en otras sentencias COBOL, como por ejemplo IF sentencias que comparan un elemento de datos alfanumérico con un elemento de datos que tiene USAGE NATIONAL.

Puede convertir explícitamente a y desde elementos de datos nacionales utilizando las funciones intrínsecas NATIONAL-OF y DISPLAY-OF, respectivamente. Utilizando estas funciones intrínsecas, puede especificar una página de códigos para la conversión que sea diferente de la página de códigos que está en vigor para un elemento de datos.

Tareas relacionadas
“Conversión de caracteres alfanuméricos, DBCS y enteros a nacionales (MOVE)” en la página 199
“Conversión de alfanuméricos o DBCS a nacional (NATIONAL-OF)” en la página 200
“Conversión de nacional a alfanumérico (DISPLAY-OF)” en la página 200
“Alteración temporal de la página de códigos predeterminada” en la página 201
“Comparación de datos nacionales (UTF-16)” en la página 205
Capítulo 11, “Establecimiento del entorno local”, en la página 213

Conversión de caracteres alfanuméricos, DBCS y enteros a nacionales (MOVE)

Puede utilizar una sentencia MOVE para convertir implícitamente datos en representación nacional.

Acerca de esta tarea

Puede mover los siguientes tipos de datos a elementos de datos de categoría nacionales o editados a nivel nacional y, por lo tanto, convertir los datos en representación nacional:

- Alfabético
- Alfanumérico
- Alfanumérico-editado
- DBCS
- Entero de USAGE DISPLAY
- Numérico-editado de USAGE DISPLAY

Del mismo modo, puede mover los siguientes tipos de datos a elementos de datos editados numéricos que tienen USAGE NATIONAL:

- Alfanumérico
- Visualizar coma flotante (coma flotante de USAGE DISPLAY)
- Numérico-editado de USAGE DISPLAY
- Entero de USAGE DISPLAY

Para obtener reglas completas sobre movimientos a datos nacionales, consulte la referencia relacionada sobre la sentencia MOVE .

Por ejemplo, la sentencia MOVE siguiente mueve el literal alfanumérico "AB" al elemento de datos nacional UTF16-Data:

```
01  UTF16-Data  Pic N(2) Usage National.
    . . .
    Move "AB" to UTF16-Data
```

Después de la sentencia MOVE anterior, UTF16-Data contiene NX'41004200', la representación nacional de los caracteres alfanuméricos 'AB'.

Si el relleno es necesario en un elemento de datos de recepción que tiene USAGE NATIONAL, se utiliza el carácter de espacio UTF-16 predeterminado (NX'2000'). Si se requiere truncamiento, se produce en el límite de una posición de carácter nacional.

Tareas relacionadas
“Asignación de valores a elementos de datos elementales (MOVE)” en la página 28
“Asignación de valores a elementos de datos de grupo (MOVE)” en la página 29
“Visualización de datos numéricos” en la página 37
“Codificación para el uso del soporte DBCS” en la página 210

Referencias relacionadas
sentencia MOVE (*COBOL for Linux en x86 Consulta de lenguaje*)

Conversión de alfanuméricos o DBCS a nacional (NATIONAL-OF)

Utilice la función intrínseca NATIONAL-OF para convertir datos alfabéticos, alfanuméricos, o DBCS a un elemento de datos nacional. Especifique la página de códigos fuente como el segundo argumento si el origen está codificado en una página de códigos diferente de la que está en vigor para el elemento de datos.

Acerca de esta tarea

“Ejemplo: conversión a y desde datos nacionales” en la página 201

Tareas relacionadas
“Proceso de datos UTF-8 utilizando tipos de datos UTF-16 (nacional)” en la página 208
“Procesando datos en chino GB 18030” en la página 209
“Procesando elementos de datos alfanuméricos que contienen datos DBCS” en la página 212

Referencias relacionadas
NATIONAL-OF (*COBOL for Linux en x86 Consulta de lenguaje*)

Conversión de nacional a alfanumérico (DISPLAY-OF)

Utilice la función intrínseca DISPLAY-OF para convertir datos nacionales en una serie de caracteres alfanuméricos (USAGE DISPLAY) que esté representada en una página de códigos que especifique como segundo argumento.

Acerca de esta tarea

Si omite el segundo argumento, la página de códigos de salida se determina a partir del entorno local de tiempo de ejecución.

Si especifica una página de códigos EBCDIC o ASCII que combina el juego de caracteres de un solo byte (SBCS) y los caracteres DBCS, la serie devuelta puede contener una mezcla de caracteres SBCS y DBCS. Las subseries DBCS están delimitadas por caracteres de desplazamiento a teclado ideográfico y de desplazamiento a teclado ideográfico si la página de códigos en vigor para la función es una página de códigos EBCDIC.

“Ejemplo: conversión a y desde datos nacionales” en la página 201

Conceptos relacionados
“El entorno local activo” en la página 213

Tareas relacionadas
“Proceso de datos UTF-8 utilizando tipos de datos UTF-16 (nacional)” en la página 208
“Procesando datos en chino GB 18030” en la página 209

Referencias relacionadas
DISPLAY-OF (*COBOL for Linux en x86 Consulta de lenguaje*)

Alteración temporal de la página de códigos predeterminada

En algunos casos, es posible que tenga que convertir datos a o desde una página de códigos que difiera de la página de códigos de que está en vigor en tiempo de ejecución. Para ello, convierta el elemento utilizando una función de conversión en la que especifique explícitamente la página de códigos.

Acerca de esta tarea

Si especifica una página de códigos como argumento en la función intrínseca DISPLAY-OF , y la página de códigos difiere de la página de códigos que está en vigor en tiempo de ejecución, no utilice el resultado de la función en ninguna operación que implique una conversión implícita (como una asignación a, o comparación con, un elemento de datos nacional). Estas operaciones presuponen la página de códigos de tiempo de ejecución de .

Ejemplo: conversión a y desde datos nacionales

El ejemplo siguiente muestra las funciones intrínsecas NATIONAL-OF y DISPLAY-OF y la sentencia MOVE para convertir a y desde elementos de datos nacionales (UTF-16). También demuestra la necesidad de conversiones explícitas cuando opera en series codificadas en varias páginas de códigos.

```
* . . .
 01  Data-in-Unicode          pic N(100) usage national.
 01  Data-in-Greek            pic X(100).
 01  other-data-in-US-English pic X(12) value "PRICE in $ =".
* . . .
     Read Greek-file into Data-in-Greek
     Move function National-of(Data-in-Greek, "ISO8859-7")
         to Data-in-Unicode
* . . . process Data-in-Unicode here . . .
     Move function Display-of(Data-in-Unicode, "ISO8859-7")
         to Data-in-Greek
     Write Greek-record from Data-in-Greek
```

El ejemplo anterior funciona correctamente porque se ha especificado la página de códigos de entrada. Data-in-Greek se convierte como datos representados en ISO8859-7 (Ascii griego). Sin embargo, la siguiente sentencia da como resultado una conversión incorrecta a menos que todos los caracteres del elemento estén entre los que tienen una representación común en las páginas de códigos en griego y en inglés:

```
Move Data-in-Greek to Data-in-Unicode
```

Suponiendo que el entorno local en vigor es en_US.ISO8859-1, la sentencia MOVE anterior convierte Data-in-Greek a Unicode basándose en la conversión de la página de códigos ISO8859-1 a UTF-16LE . Esta conversión no produce los resultados esperados porque Data-in-Greek está codificado en ISO8859-7.

Si establece el entorno local en el_GR.ISO8859-7 (es decir, el programa maneja datos ASCII en griego), puede codificar el mismo ejemplo correctamente como se indica a continuación:

```
* . . .
 01  Data-in-Unicode pic N(100) usage national.
 01  Data-in-Greek   pic X(100).
* . . .
     Read Greek-file into Data-in-Greek
* . . . process Data-in-Greek here ...
* . . . or do the following (if need to process data in Unicode):
     Move Data-in-Greek to Data-in-Unicode
* . . . process Data-in-Unicode
     Move function Display-of(Data-in-Unicode) to Data-in-Greek
     Write Greek-record from Data-in-Greek
```

Tareas relacionadas
Capítulo 11, “Establecimiento del entorno local”, en la página 213

Utilización de grupos nacionales

Para definir un elemento de datos de grupo como un grupo nacional, codifique una cláusula GROUP-USAGE NATIONAL a nivel de grupo para el elemento. El grupo sólo puede contener elementos de datos que tengan explícita o implícitamente USAGE NATIONAL.

Acerca de esta tarea

La siguiente entrada de descripción de datos especifica que un grupo level-01 y sus grupos subordinados son elementos de grupo nacionales:

```
01  Nat-Group-1    GROUP-USAGE NATIONAL.
    02  Group-1.
        04  Month    PIC 99.
        04  DayOf    PIC 99.
        04  Year     PIC 9999.
    02  Group-2    GROUP-USAGE NATIONAL.
        04  Amount   PIC 9(4).99  USAGE NATIONAL.
```

En el ejemplo anterior, Nat-Group-1 es un grupo nacional y sus grupos subordinados Group-1 y Group-2 también son grupos nacionales. Una cláusula GROUP-USAGE NATIONAL está implícita para Group-1y USAGE NATIONAL está implícita para los elementos subordinados en Group-1. Month, DayOfy Year son elementos decimales nacionales y Amount es un elemento editado numérico que tiene USAGE NATIONAL.

Puede subordinar grupos nacionales dentro de grupos alfanuméricos como en el ejemplo siguiente:

```
01  Alpha-Group-1.
    02  Group-1.
        04  Month   PIC 99.
        04  DayOf   PIC 99.
        04  Year    PIC 9999.
    02  Group-2   GROUP-USAGE NATIONAL.
        04  Amount  PIC 9(4).99.
```

En el ejemplo anterior, Alpha-Group-1 y Group-1 son grupos alfanuméricos; USAGE DISPLAY está implícito para los elementos subordinados en Group-1. (Si Alpha-Group-1 se especifica USAGE NATIONAL a nivel de grupo, USAGE NATIONAL estaría implícito para cada uno de los elementos subordinados en Group-1. Sin embargo, Alpha-Group-1 y Group-1 serían grupos alfanuméricos, no grupos nacionales, y se comportarían como grupos alfanuméricos durante operaciones como movimientos y comparaciones.) Group-2 es un grupo nacional y USAGE NATIONAL está implícito para el elemento editado numérica-editado Amount.

No puede subordinar grupos alfanuméricos dentro de grupos nacionales. Todos los elementos elementales de un grupo nacional deben describirse explícita o implícitamente como USAGE NATIONAL,

y todos los elementos de grupo de un grupo nacional deben describirse explícita o implícitamente como GROUP-USAGE NATIONAL.

Conceptos relacionados
"Grupos nacionales" en la página 198

Tareas relacionadas
"Utilización de grupos nacionales como elementos elementales" en la página 203
"Utilización de grupos nacionales como elementos de grupo" en la página 203

Referencias relacionadas
cláusula GROUP-USAGE (*COBOL for Linux en x86 Consulta de lenguaje*)

Utilización de grupos nacionales como elementos elementales

En la mayoría de los casos, puede utilizar un grupo nacional como si fuera un elemento de datos elemental.

Acerca de esta tarea

En el ejemplo siguiente, un elemento de grupo nacional, Group-1, se mueve a un elemento editado a nivel nacional, Edited-date. Puesto que Group-1 se trata como un elemento de datos elemental durante el traslado, la edición tiene lugar en el elemento de datos receptor. El valor de Edited-date después del movimiento es 06/23/2010 en caracteres nacionales.

```
01  Edited-date  PIC NN/NN/NNNN  USAGE NATIONAL.
01  Group-1    GROUP-USAGE NATIONAL.
    02 Month     PIC 99   VALUE 06.
    02 DayOf     PIC 99   VALUE 23.
    02 Year      PIC 9999 VALUE 2010.
    . . .
    MOVE Group-1 to Edited-date.
```

Si Group-1 fuera en cambio un grupo alfanumérico en el que cada uno de sus elementos subordinados tuviera USAGE NATIONAL (especificado explícitamente con una cláusula USAGE NATIONAL en cada elemento elemental, o implícitamente con una cláusula USAGE NATIONAL a nivel de grupo), se produciría un movimiento de grupo, en lugar de un movimiento elemental. Ni la edición ni la conversión tendrían lugar durante el movimiento. El valor de las primeras ocho posiciones de caracteres de Edited-date después del movimiento sería 06232010 en caracteres nacionales, y el valor de las dos posiciones de caracteres restantes sería 4 bytes de espacios alfanuméricos.

Tareas relacionadas
"Asignación de valores a elementos de datos de grupo (MOVE)" en la página 29
"Comparación de datos nacionales y operandos de grupos alfanuméricos" en la página 208
"Utilización de grupos nacionales como elementos de grupo" en la página 203

Referencias relacionadas
sentencia MOVE (*COBOL for Linux en x86 Consulta de lenguaje*)

Utilización de grupos nacionales como elementos de grupo

En algunos casos, cuando se utiliza un grupo nacional, se maneja con semántica de grupo; es decir, los elementos elementales del grupo se reconocen o procesan.

Acerca de esta tarea

En el ejemplo siguiente, una sentencia INITIALIZE que actúa sobre el elemento de grupo nacional Group-OneN hace que el valor 15 en caracteres nacionales se mueva sólo a los elementos numéricos del grupo:

```
01  Group-OneN    Group-Usage National.
    05  Trans-codeN     Pic N   Value "A".
    05  Part-numberN    Pic NN  Value "XX".
    05  Trans-quanN     Pic 99  Value 10.
    . . .
    Initialize Group-OneN Replacing Numeric Data By 15
```

Puesto que sólo `Trans-quanN` en `Group-OneN` anterior es numérico, sólo `Trans-quanN` recibe el valor 15. Los otros elementos subordinados no se modifican.

La tabla siguiente resume los casos en los que los grupos nacionales se procesan con semántica de grupo.

Tabla 18. ***Elementos de grupo nacionales que se procesan con semántica de grupo***

Característica de idioma	**Usos de los grupos nacionales**	**Comentario**
Frase `CORRESPONDING` de la sentencia `ADD`, `SUBTRACTo MOVE`	Especifique un elemento de grupo nacional para procesarlo como un grupo de acuerdo con las reglas de la frase `CORRESPONDING` .	Los elementos elementales del grupo nacional se procesan como elementos elementales que tienen `USAGE NATIONAL` dentro de un grupo alfanumérico.
Sentencia `INITIALIZE`	Especifique un grupo nacional para procesarlo como grupo de acuerdo con las reglas de la sentencia `INITIALIZE` .	Los elementos elementales dentro del grupo nacional se inicializan como elementos elementales que tienen `USAGE NATIONAL` dentro de un grupo alfanumérico.
Cualificación de nombre	Utilice el nombre de un elemento de grupo nacional para calificar los nombres de los elementos de datos elementales y de los elementos de grupo subordinados en el grupo nacional.	Siga las mismas reglas para la cualificación que para un grupo alfanumérico.
Frase `THROUGH` de la cláusula `RENAMES`	Para especificar un elemento de grupo nacional en la frase `THROUGH` , utilice las mismas reglas que para un elemento de grupo alfanumérico.	El resultado es un elemento de grupo alfanumérico.
Frase `FROM` de la sentencia `XML GENERATE`	Especifique un elemento de grupo nacional en la frase `FROM` para procesarlo como un grupo de acuerdo con las reglas de la sentencia `XML GENERATE` .	Los elementos elementales del grupo nacional se procesan como elementos elementales que tienen `USAGE NATIONAL` dentro de un grupo alfanumérico.

Tareas relacionadas
“Inicialización de una estructura (INITIALIZE)” en la página 27
“Inicialización de una tabla (INITIALIZE)” en la página 65
“Asignación de valores a elementos de datos elementales (MOVE)” en la página 28
“Asignación de valores a elementos de datos de grupo (MOVE)” en la página 29
“Búsqueda de la longitud de los elementos de datos” en la página 112
“Generando salida XML” en la página 439

Referencias relacionadas
Cualificación (*COBOL for Linux en x86 Consulta de lenguaje*)
Cláusula RENAMES (*COBOL for Linux en x86 Consulta de lenguaje*)

Almacenamiento de datos de caracteres

Utilice la tabla siguiente para comparar la codificación alfanumérica (DISPLAY), DBCS (DISPLAY-1) y Unicode (NATIONAL) y para planificar el uso del almacenamiento.

Tabla 19. ***Codificación y tamaño de datos alfanuméricos, DBCS y nacionales***

Características	**DISPLAY**	**DISPLAY-1**	**NATIONAL**
Unidad de codificación de caracteres	1 byte	2 bytes	2 bytes
Página de códigos	ASCII, EUC, UTF-8, o EBCDIC[3]	ASCII DBCS o EBCDIC DBCS[3]	UTF-16LE[1]
Unidades de codificación por carácter gráfico	1	1	1 o 2[2]
Bytes por carácter gráfico	1 byte	2 bytes	2 o 4 bytes

1. Los literales nacionales del programa fuente se convierten a UTF-16 para su uso en tiempo de ejecución.
2. La mayoría de los caracteres se representan en UTF-16 utilizando una unidad de codificación. En concreto, los caracteres siguientes se representan utilizando una única unidad de codificación UTF-16 por carácter:
   - Caracteres COBOL A-Z, a-z, 0-9, espacio, +-*/= $,;. " () > <: "
   - Todos los caracteres que se convierten de una página de códigos EBCDIC, ASCII o EUC
3. En función del entorno local, la opción `CHAR(NATIVE)` o `CHAR(EBCDIC)` y los valores de la variable de entorno EBCDIC_CODEPAGE

Conceptos relacionados
"Unicode y la codificación de caracteres de idioma" en la página 190

Tareas relacionadas
"Especificación de la página de códigos para datos de caracteres" en la página 214

Referencias relacionadas
"char" en la página 272

Comparación de datos nacionales (UTF-16)

Puede comparar datos nacionales de (UTF-16), es decir, literales nacionales y elementos de datos que tienen `USAGE NATIONAL` (ya sea de clase nacional o de clase numérica), explícita o implícitamente con otros tipos de datos en condiciones de relación.

Acerca de esta tarea

Puede codificar expresiones condicionales que utilicen datos nacionales en las sentencias siguientes:

- `EVALUATE`
- `IF`
- `INSPECT`
- `PERFORM`
- `SEARCH`
- `STRING`
- `UNSTRING`

Para obtener detalles completos sobre la comparación de elementos de datos nacionales con otros elementos de datos, consulte las referencias relacionadas.

Tareas relacionadas
“Comparación de dos operandos nacionales de clase” en la página 206
“Comparación de operandos nacionales y numéricos de clase” en la página 207
“Comparación de operandos numéricos nacionales y otros numéricos” en la página 207
“Comparación de operandos de serie de caracteres nacionales y otros” en la página 207
“Comparación de datos nacionales y operandos de grupos alfanuméricos” en la página 208

Referencias relacionadas
Condiciones de relación (*COBOL for Linux en x86 Consulta de lenguaje*)
Condiciones de relación generales (*COBOL for Linux en x86 Consulta de lenguaje*)
Comparaciones nacionales (*COBOL for Linux en x86 Consulta de lenguaje*)
Comparaciones de grupo (*COBOL for Linux en x86 Consulta de lenguaje*)

Comparación de dos operandos nacionales de clase

Puede comparar los valores de caracteres de dos operandos de clase nacional.

Acerca de esta tarea

Cualquiera de los operandos (o ambos) puede ser cualquiera de los siguientes tipos de elementos:

- Un grupo nacional
- Un elemento de datos de categoría elemental editado nacional o nacional
- Un elemento de datos editado numérica-que tiene `USAGE NATIONAL`

Uno de los operandos puede ser, en cambio, un literal nacional o una función intrínseca nacional.

Utilice la opción de compilador `NCOLLSEQ` para determinar qué tipo de comparación se debe realizar:

`NCOLLSEQ(BINARY)`
Cuando compara dos operandos nacionales de clase de la misma longitud, se determina que son iguales si todos los pares de los caracteres correspondientes son iguales. De lo contrario, la comparación de los valores binarios del primer par de caracteres desiguales determina el operando con el valor binario más grande.

Cuando compara operandos de longitudes desiguales, el operando más corto se trata como si se rellenara a la derecha con caracteres de espacio UTF-16 predeterminados (`NX'2000'`) a la longitud del operando más largo.

`NCOLLSEQ(LOCALE)`
Cuando se utiliza una comparación basada en el entorno local, los operandos se comparan utilizando el algoritmo para el orden de clasificación que está asociado con el entorno local en vigor. Los espacios finales se truncan desde los operandos, excepto que un operando que consta de todos los espacios se trunca en un único espacio.

Cuando compara operandos de longitudes desiguales, el operando más corto no se amplía con espacios porque una extensión de este tipo podría alterar los resultados esperados para el entorno local.

La cláusula `PROGRAM COLLATING SEQUENCE` no afecta a la comparación de dos operandos nacionales de clase .

Conceptos relacionados
“Grupos nacionales” en la página 198

Tareas relacionadas
“Utilización de grupos nacionales” en la página 202

Referencias relacionadas
"NCOLLSEQ" en la página 290
Comparaciones nacionales (*COBOL for Linux en x86 Consulta de lenguaje*)

Comparación de operandos nacionales y numéricos de clase

Puede comparar literales nacionales o elementos de datos nacionales de clase con literales enteros o elementos de datos numéricos que están definidos como enteros (es decir, elementos decimales nacionales o elementos decimales con zona). Como máximo, uno de los operandos puede ser un literal.

Acerca de esta tarea

También puede comparar literales nacionales o elementos de datos nacionales de clase con elementos de datos de coma flotante (es decir, visualizar elementos de coma flotante o de coma flotante nacional).

Los operandos numéricos se convierten a representación nacional (UTF-16) si todavía no están en representación nacional. Se realiza una comparación de los valores de caracteres nacionales de los operandos.

Referencias relacionadas
Condiciones de relación generales (*COBOL for Linux en x86 Consulta de lenguaje*)

Comparación de operandos numéricos nacionales y otros numéricos

Los operandos numéricos nacionales (operandos decimales nacionales y de coma flotante nacional) son elementos de datos de clase numérica que tienen `USAGE NATIONAL`.

Acerca de esta tarea

Puede comparar los valores algebraicos de operandos numéricos independientemente de su USAGE. Por lo tanto, puede comparar un elemento decimal nacional o un elemento de coma flotante nacional con un elemento binario, un elemento decimal interno, un elemento decimal con zona, un elemento de coma flotante de visualización o cualquier otro elemento numérico.

Tareas relacionadas
"Definición de elementos de datos numéricos nacionales" en la página 197

Referencias relacionadas
Condiciones de relación generales (*COBOL for Linux en x86 Consulta de lenguaje*)

Comparación de operandos de serie de caracteres nacionales y otros

Puede comparar el valor de carácter de un literal nacional o elemento de datos nacional de clase con el valor de carácter de cualquiera de los siguientes operandos de serie de caracteres: alfabético, alfanumérico, editado alfanumérico, DBCS, o numérico-editado de `USAGE DISPLAY`.

Acerca de esta tarea

Estos operandos se tratan como si se trasladaran a un elemento de datos nacional elemental. Los caracteres se convierten a representación nacional (UTF-16) y la comparación continúa con dos operandos de caracteres nacionales.

Tareas relacionadas
"Utilización de constantes figurativas de carácter nacional" en la página 196
"Comparación de literales DBCS" en la página 211

Referencias relacionadas
Comparaciones nacionales (*COBOL for Linux en x86 Consulta de lenguaje*)

Comparación de datos nacionales y operandos de grupos alfanuméricos

Puede comparar un literal nacional, un elemento de grupo nacional o cualquier elemento de datos elemental que tenga USAGE NATIONAL con un grupo alfanumérico.

Acerca de esta tarea

No se convierte ninguno de los operandos. El operando nacional se trata como si se moviera a un elemento de grupo alfanumérico del mismo tamaño en bytes que el operando nacional, y se comparan los dos grupos. Se realiza una comparación alfanumérica independientemente de la representación de los elementos subordinados en el operando de grupo alfanumérico.

Por ejemplo, Group-XN es un grupo alfanumérico que consta de dos elementos subordinados que tienen USAGE NATIONAL:

```
01  Group-XN.
    02 TransCode PIC NN   Value "AB"  Usage National.
    02 Quantity  PIC 999  Value 123   Usage National.
    . . .
    If N"AB123" = Group-XN  Then Display "EQUAL"
    Else Display "NOT EQUAL".
```

Cuando se ejecuta la sentencia IF anterior, los 10 bytes del literal nacional N"AB123" se comparan byte a byte con el contenido de Group-XN. Los elementos se comparan por igual y se muestra "EQUAL".

Referencias relacionadas
Comparaciones de grupo (*COBOL for Linux en x86 Consulta de lenguaje*)

Proceso de datos UTF-8 utilizando tipos de datos UTF-16 (nacional)

Para procesar datos UTF-8 , en primer lugar, convierta los datos UTF-8 a UTF-16 en un elemento de datos nacional. Después de procesar los datos nacionales, vuelva a convertirlos a UTF-8 para la salida. Para las conversiones, utilice las funciones intrínsecas NATIONAL-OF y DISPLAY-OF, respectivamente. Utilice la página de códigos 1208 para los datos UTF-8 .

Acerca de esta tarea

Como alternativa al método recomendado para procesar datos UTF-8 utilizando

USAGE UTF-8

elementos de datos, también puede procesar datos UTF-8 almacenándolos en elementos de datos alfanuméricos y, a continuación, convirtiéndolos a UTF-16 en un elemento de datos nacional.

Realice los pasos siguientes para convertir datos ASCII o EBCDIC a UTF-8 (a menos que la página de códigos del entorno local en vigor sea UTF-8, en cuyo caso los datos alfanuméricos nativos ya están codificados en UTF-8):

Procedimiento

1. Utilice la función NATIONAL-OF para convertir la serie ASCII o EBCDIC en una serie nacional (UTF-16).
2. Utilice la función DISPLAY-OF para convertir la serie nacional a UTF-8.

Resultados

El ejemplo siguiente convierte los datos EBCDIC griegos a UTF-8:

```
01 Greek-EBCDIC pic X(10) value "αβγδεζηθ".
01 UnicodeString pic N(10).
01 UTF-8-String pic X(20).
    Move function National-of(Greek-EBCDIC, 00875) to UnicodeString
    Move function Display-of(UnicodeString, 01208) to UTF-8-String
```

Nota de uso: tenga cuidado si utiliza la modificación de referencia para hacer referencia a datos codificados en UTF-8. Los caracteres UTF-8 se codifican con un número variable de bytes por carácter. Evite operaciones que puedan dividir un carácter de varios bytes.

Tareas relacionadas
"Referencia a subseries de elementos de datos" en la página 101
"Conversión a o desde representación nacional (Unicode)" en la página 199
"Análisis de documentos XML codificados en UTF-8" en la página 429

Procesando datos en chino GB 18030

GB 18030 es un estándar de carácter nacional especificado por el gobierno de la República Popular China.

Acerca de esta tarea

COBOL para Linux admite GB 18030. Si la página de códigos especificada para el entorno local en vigor es GB18030 (una página de códigos que admite GB 18030), USAGE DISPLAY los elementos de datos que contienen GB 18030 caracteres codificados en GB18030 se pueden procesar en un programa. Los caracteres GB 18030 toman de 1 a 4 bytes cada uno. Por lo tanto, la lógica del programa debe ser sensible a la naturaleza multibyte de los datos.

Puede procesar GB 18030 caracteres de estas maneras:

- Utilice elementos de datos nacionales para definir y procesar GB 18030 caracteres representados en UTF-16, CCSID 01200.
- Procese datos en cualquier página de códigos (incluido GB18030, que tiene CCSID 1392) convirtiendo los datos a UTF-16, procesando los datos UTF-16 y, a continuación, convirtiendo los datos de nuevo a la representación de página de códigos original.

Cuando necesite procesar datos en chino GB 18030 que requieran conversión, primero convierta los datos de entrada a UTF-16 en un elemento de datos nacional. Después de procesar el elemento de datos nacional, conviértalo de nuevo al chino GB 18030 para la salida. Para las conversiones, utilice las funciones intrínsecas NATIONAL-OF y DISPLAY-OF, respectivamente, y especifique GB18030 o 1392 como segundo argumento de cada función.

El ejemplo siguiente ilustra estas conversiones:

```
 01 Chinese-ASCII pic X(16) value "奥林匹克运动会".
 01 Chinese-GB18030-String pic X(16).
 01 UnicodeString pic N(14).
 . . .
     Move function National-of(Chinese-ASCII, 1392) to UnicodeString
*  Process data in Unicode
     Move function Display-of(UnicodeString, 1392) to Chinese-GB18030-String
```

Tareas relacionadas
"Conversión a o desde representación nacional (Unicode)" en la página 199
"Codificación para el uso del soporte DBCS" en la página 210

Referencias relacionadas
"Almacenamiento de datos de caracteres" en la página 205

Codificación para el uso del soporte DBCS

IBM COBOL for Linux en x86 da soporte a la utilización de aplicaciones en cualquiera de los muchos idiomas nacionales, incluidos los idiomas que utilizan juegos de caracteres de doble byte (DBCS).

Acerca de esta tarea

La lista siguiente resume el soporte para DBCS:

- Caracteres DBCS en palabras definidas por el usuario (nombresmultibyte)
- Caracteres DBCS en comentarios
- Elementos de datos DBCS (definidos con PICTURE N, Go G y B)
- Literales DBCS
- Orden de clasificación
- Opción de compilador SOSI
- variable de entorno DBCS_CODEPAGE

Tareas relacionadas
“Definición de datos DBCS” en la página 210
“Utilización de literales DBCS” en la página 211
“Prueba de caracteres DBCS válidos” en la página 212
“Procesando elementos de datos alfanuméricos que contienen datos DBCS” en la página 212
Capítulo 11, “Establecimiento del entorno local”, en la página 213
“Control de la secuencia de clasificación con un entorno local” en la página 219

Referencias relacionadas
“SOSI” en la página 295

Definición de datos DBCS

Utilice las cláusulas PICTURE y USAGE para definir elementos de datos DBCS. Los elementos de datos DBCS pueden utilizar PICTURE símbolos G, G y B, o N. Cada posición de carácter DBCS tiene una longitud de 2 bytes.

Acerca de esta tarea

Puede especificar un elemento de datos DBCS utilizando la cláusula USAGE DISPLAY-1 . Cuando utilice PICTURE symbol G, debe especificar USAGE DISPLAY-1. Cuando se utiliza el símbolo PICTURE N pero se omite la cláusula USAGE , USAGE DISPLAY-1 o USAGE NATIONAL está implícito en función del valor de la opción de compilador NSYMBOL .

Si utiliza una cláusula VALUE con la cláusula USAGE en la definición de un elemento DBCS, debe especificar un literal DBCS o la constante figurativa SPACE o SPACES.

Si un elemento de datos tiene USAGE DISPLAY-1 (explícita o implícitamente), el entorno local seleccionado debe indicar una página de códigos que incluya caracteres DBCS. Si la página de códigos del entorno local no incluye caracteres DBCS, estos elementos de datos se marcan como errores.

A efectos del manejo de modificaciones de referencia, se considera que cada carácter de un elemento de datos DBCS ocupa el número de bytes que corresponde a la anchura de página de códigos (es decir, 2).

Tareas relacionadas
Capítulo 11, “Establecimiento del entorno local”, en la página 213

Referencias relacionadas
“Entornos locales y páginas de códigos soportadas” en la página 217
“SÍMBOLO” en la página 290

Utilización de literales DBCS

Puede utilizar el prefijo N o G para representar un literal DBCS.

Acerca de esta tarea

Es decir, puede especificar un literal DBCS de una de estas formas:

- N'*caracteres dbcs*' (siempre que la opción de compilador NSYMBOL(DBCS) esté en vigor)
- G'*caracteres dbcs*'

Puede utilizar comillas (") o apóstrofos (') como delimitadores de un literal DBCS independientemente de el valor de la opción de compilador APOST o QUOTE . Debe codificar el mismo delimitador de apertura y cierre para un literal DBCS.

Si la opción de compilador SOSI está en vigor, el carácter de control de desplazamiento a teclado ideográfico (SO) X'1E' debe ir inmediatamente después del delimitador de apertura y el carácter de control de desplazamiento a teclado estándar (SI) X'1F' debe ir inmediatamente antes del delimitador de cierre.

Además de los literales DBCS, puede utilizar literales alfanuméricos para especificar cualquier carácter en una de las páginas de códigos soportadas. Sin embargo, Si la opción de compilador SOSI está en vigor, cualquier serie de caracteres DBCS que esté dentro de un literal alfanumérico debe estar delimitada por los caracteres SO y SI.

No puede continuar un literal alfanumérico que contenga multibyte caracteres. La longitud de un literal DBCS está igualmente limitada por el espacio disponible en el Área B en una sola línea de origen. Por lo tanto, la longitud máxima de un literal DBCS es de 28 caracteres de doble byte.

Un literal alfanumérico que contiene multibyte caracteres se procesa byte por byte, es decir, con la semántica adecuada para los caracteres de un solo byte, excepto cuando se convierte explícita o implícitamente en una representación de datos nacional, como por ejemplo en una asignación o comparación con un elemento de datos nacional.

Tareas relacionadas
“Comparación de literales DBCS” en la página 211
“Utilización de constantes figurativas” en la página 22

Referencias relacionadas
“SÍMBOLO” en la página 290
“SOSI” en la página 295
literales DBCS (*COBOL for Linux en x86 Consulta de lenguaje*)

Comparación de literales DBCS

Las comparaciones de literales DBCS se basan en el entorno local de tiempo de compilación. Por lo tanto, no utilice literales DBCS dentro de una sentencia que exprese una condición relacional implícita entre dos literales DBCS (como VALUE G'*literal-1*' THRU G'*literal-2*') a menos que el entorno local de tiempo de ejecución previsto sea el mismo que el entorno local de tiempo de compilación.

Acerca de esta tarea

Tareas relacionadas
“Comparación de datos nacionales (UTF-16)” en la página 205
Capítulo 11, “Establecimiento del entorno local”, en la página 213

Referencias relacionadas
“COLLSEQ” en la página 275
Literales DBCS (*COBOL for Linux en x86 Consulta de lenguaje*)
Comparaciones DBCS (*COBOL for Linux en x86 Consulta de lenguaje*)

Prueba de caracteres DBCS válidos

La prueba de clase Kanji comprueba si hay caracteres gráficos en japonés válidos. Esta prueba incluye los juegos de caracteres Katakana, Hiragana, Roman y Kanji.

Acerca de esta tarea

Las pruebas de clase Kanji y DBCS se definen para que sean coherentes con sus definiciones de IBM Z . Ambas pruebas de clase se realizan internamente convirtiendo los caracteres de doble byte a los caracteres de doble byte definidos para z/OS. Los caracteres de doble byte convertidos se prueban para los caracteres gráficos DBCS y japonés.

La prueba de clase Kanji se realiza comprobando los caracteres convertidos para el rango `X'41'` a `X'7E'` en el primer byte y `X'41'` a `X'FE'` en el segundo byte, más el carácter de espacio `X'4040'`.

La prueba de clase DBCS comprueba si hay caracteres gráficos válidos para la página de códigos.

La prueba de clase DBCS se realiza comprobando los caracteres convertidos para el rango `X'41'` a `X'FE'` en el primer y segundo byte de cada carácter, más el carácter de espacio `X'4040'`.

Tareas relacionadas
“Codificación de expresiones condicionales” en la página 87

Referencias relacionadas
Condición de clase (*COBOL for Linux en x86 Consulta de lenguaje*)

Procesando elementos de datos alfanuméricos que contienen datos DBCS

Si utiliza operaciones orientadas a bytes (por ejemplo, `STRING`, `UNSTRING`o modificación de referencia) en un elemento de datos alfanumérico que contiene caracteres DBCS, los resultados son imprevisibles. En su lugar, debe convertir el elemento en un elemento de datos nacional antes de procesarlo.

Acerca de esta tarea

Es decir, siga estos pasos:

Procedimiento

1. Convierta el elemento a UTF-16 en un elemento de datos nacional utilizando una sentencia `MOVE` o la función intrínseca `NATIONAL-OF` .
2. Procese el elemento de datos nacional según sea necesario.
3. Vuelva a convertir el resultado en un elemento de datos alfanumérico utilizando la función intrínseca `DISPLAY-OF` .

Resultados

Tareas relacionadas
“Unión de elementos de datos (STRING)” en la página 95
“División de elementos de datos (UNSTRING)” en la página 97
“Referencia a subseries de elementos de datos” en la página 101
“Conversión a o desde representación nacional (Unicode)” en la página 199

Capítulo 11. Establecimiento del entorno local

Puede escribir aplicaciones para reflejar los convenios culturales del entorno local que está en vigor cuando se ejecutan las aplicaciones. Las convenciones culturales incluyen orden de clasificación, clasificación de caracteres e idioma nacional; y formatos de fechas y horas, números, unidades monetarias, direcciones postales y números de teléfono.

Acerca de esta tarea

Con COBOL para Linux, puede seleccionar las páginas de códigos y secuencias de clasificación adecuadas, y puede utilizar elementos de lenguaje y opciones de compilador para manejar Unicode, juegos de caracteres de un solo byte y juegos de caracteres de doble byte (DBCS).

Conceptos relacionados
"El entorno local activo" en la página 213

Tareas relacionadas
"Especificación de la página de códigos para datos de caracteres" en la página 214
"Utilización de variables de entorno para especificar un entorno local" en la página 215
"Control de la secuencia de clasificación con un entorno local" en la página 219
"Acceso al entorno local activo y a los valores de página de códigos" en la página 223

El entorno local activo

Un *entorno local* es una colección de datos que codifica información sobre un entorno cultural. El entorno local activo es el entorno local que está en vigor al compilar o ejecutar el programa. Puede establecer un entorno cultural para una aplicación especificando el entorno local activo.

Sólo un entorno local puede estar activo a la vez.

El entorno local activo afecta al comportamiento de estas interfaces culturalmente sensibles para todo el programa:

- Páginas de códigos utilizadas para datos de caracteres
- Mensajes
- Orden de clasificación
- Formatos de fecha y hora
- Clasificación de caracteres y conversión de mayúsculas/minúsculas

El entorno local activo no afecta a los siguientes elementos, para los que 85 COBOL Estándar define un idioma y un comportamiento específicos:

- Separadores de coma decimal y agrupación
- Signo de moneda

El entorno local activo determina la página de códigos para compilar y ejecutar programas:

- La página de códigos que se utiliza para la compilación se basa en el valor de entorno local en el tiempo de compilación.
- La página de códigos que se utiliza para ejecutar una aplicación se basa en el valor de entorno local en tiempo de ejecución.

La evaluación de valores literales en el programa fuente se maneja con el entorno local que está activo durante la compilación. Por ejemplo, la conversión de literales nacionales de la representación de origen a UTF-16 para ejecutar el programa utiliza el entorno local de tiempo de compilación.

COBOL para Linux determina el valor del entorno local activo a partir de una combinación de las variables de entorno aplicables y los valores del sistema. Las variables de entorno se utilizan en primer lugar.

Si las variables de entorno no definen una categoría de entorno local aplicable, COBOL utiliza valores predeterminados y valores del sistema.

Conceptos relacionados
"Determinación del entorno local a partir de los valores del sistema" en la página 216

Tareas relacionadas
"Especificación de la página de códigos para datos de caracteres" en la página 214
"Utilización de variables de entorno para especificar un entorno local" en la página 215
"Control de la secuencia de clasificación con un entorno local" en la página 219

Referencias relacionadas
"Tipos de mensajes para los que hay traducciones disponibles" en la página 216

Especificación de la página de códigos para datos de caracteres

En un programa fuente, puede utilizar los caracteres que están representados en una página de códigos soportada en nombres COBOL, literales y comentarios. En tiempo de ejecución, puede utilizar los caracteres que se representan en una página de códigos soportada en los elementos de datos descritos con `USAGE DISPLAY`, `USAGE DISPLAY-1`o `USAGE NATIONAL`.

Acerca de esta tarea

La página de códigos que está en vigor para un elemento de datos determinado depende de los aspectos siguientes:

- Qué cláusula USAGE ha utilizado
- Si ha utilizado la frase `NATIVE` con la cláusula USAGE
- Si ha utilizado la opción de compilador `CHAR(NATIVE)` o `CHAR(EBCDIC)`
- El valor de la variable de entorno EBCDIC_CODEPAGE
- El valor de la variable de entorno DBCS_CODEPAGE
- Qué entorno local está activo

Para los elementos de datos de `USAGE NATIONAL` , la página de códigos toma como valor predeterminado UTF-16 en formato little-endian .

Para los elementos de datos de `USAGE DISPLAY` , COBOL para Linux elige entre las páginas de códigos ASCII, UTF-8, EUC, y EBCDIC de la forma siguiente:

- Los elementos de datos que se describen con la frase `NATIVE` en la cláusula USAGE o que se compilan con la opción `CHAR(NATIVE)` en vigor se codifican en un ASCII, EUC, o UTF-8 .
- Los elementos de datos que se describen sin la frase `NATIVE` en la cláusula USAGE y que se compilan con la opción `CHAR(EBCDIC)` en vigor se codifican en una página de códigos EBCDIC.

Para los elementos de datos de `USAGE DISPLAY-1` , COBOL para Linux elige entre las páginas de códigos ASCII y EBCDIC de la forma siguiente:

- Los elementos de datos que se describen con la frase `NATIVE` en la cláusula USAGE o que se compilan con la opción `CHAR(NATIVE)` en vigor se codifican en una página de códigos DBCS ASCII.
- Los elementos de datos que se describen sin la frase `NATIVE` en la cláusula USAGE y que se compilan con la opción `CHAR(EBCDIC)` en vigor se codifican en una página de códigos DBCS EBCDIC.

COBOL determina la página de códigos adecuada de la forma siguiente:

ASCII, UTF-8, EUC
Desde el entorno local activo en tiempo de ejecución

DBCS ASCII
Desde la variable de entorno DBCS_CODEPAGE, si se ha establecido, de lo contrario, la página de códigos DBCS predeterminada del valor local actual

EBCDIC
Desde la variable de entorno EBCDIC_CODEPAGE, si se ha establecido, de lo contrario, la página de códigos EBCDIC predeterminada del valor de entorno local actual

Tareas relacionadas
"Utilización de variables de entorno para especificar un entorno local" en la página 215

Referencias relacionadas
"Entornos locales y páginas de códigos soportadas" en la página 217
"Variables de entorno de ejecución" en la página 234
"char" en la página 272
Palabras COBOL con caracteres de un solo byte
(*COBOL for Linux en x86 Consulta de lenguaje*)
Palabras definidas por el usuario con caracteres de varios bytes
(*COBOL for Linux en x86 Consulta de lenguaje*)

Utilización de variables de entorno para especificar un entorno local

Utilice cualquiera de varias variables de entorno para proporcionar la información de entorno local para un programa COBOL.

Acerca de esta tarea

Para especificar una página de códigos que se utilizará para todas las categorías de entorno local (mensajes, secuencia de clasificación, formatos de fecha y hora, clasificación de caracteres y conversión de mayúsculas y minúsculas), utilice LC_ALL.

Para establecer el valor para una categoría de entorno local específica, utilice la variable de entorno adecuada:

- Utilice LC_MESSAGES para especificar el formato de las respuestas afirmativas y negativas. También puede utilizarlo para determinar si los mensajes (por ejemplo, los mensajes de error y las cabeceras de listado) están en inglés de Estados Unidos o en japonés. Para cualquier entorno local que no sea japonés, se utiliza el inglés de Estados Unidos.
- Utilice LC_COLLATE para especificar el orden de clasificación en vigor para comparaciones mayor que o menor que, por ejemplo, en condiciones de relación o en las sentencias SORT y MERGE .
- Utilice LC_TIME para especificar el formato de la fecha y hora que se muestra en los listados del compilador. Todos los demás valores de fecha y hora se controlan mediante sintaxis de lenguaje COBOL.
- Utilice LC_CTYPE para especificar la clasificación de caracteres, la conversión de mayúsculas y minúsculas y otros atributos de caracteres.

Cualquier categoría de entorno local que no haya sido especificada por una de las variables de entorno local anteriores se establece a partir del valor de la variable de entorno LANG.

Para establecer las variables de entorno local, utilice un mandato con el formato siguiente (`.`*`codepageID`* es opcional):

```
export LC_xxxx=ll_CC.codepageID
```

Aquí LC_*xxxx* es el nombre de la categoría de entorno local, *ll* es un código de idioma de dos letras en minúsculas, *CC* es un código de país ISO de dos letras en mayúsculas, y *ID de página de códigos* es la página de códigos que se utilizará para los datos nativos de DISPLAY y DISPLAY-1 . COBOL para Linux utiliza los convenios de entorno local definidos por POSIX.

Por ejemplo, para establecer el entorno local en francés canadiense codificado en ISO 8859-1, emita este mandato en la ventana de mandatos desde la que compile y ejecute una aplicación COBOL:

```
export LC_ALL=fr_CA.iso88591
```

Debe codificar un valor válido para el nombre de entorno local (*ll_CC*) y la página de códigos (*codepageID*) que especifique debe ser válida para el nombre de entorno local. Los valores válidos se muestran en la tabla de entornos locales soportados y páginas de códigos a las que se hace referencia a continuación.

Conceptos relacionados
"Determinación del entorno local a partir de los valores del sistema" en la página 216

Tareas relacionadas
"Especificación de la página de códigos para datos de caracteres" en la página 214

Referencias relacionadas
"Entornos locales y páginas de códigos soportadas" en la página 217
"Variables de entorno de compilador y tiempo de ejecución" en la página 230

Determinación del entorno local a partir de los valores del sistema

Si COBOL para Linux no puede determinar el valor de una categoría de entorno local aplicable a partir de las variables de entorno, utiliza los valores predeterminados.

Cuando los códigos de idioma y de país se determinan a partir de variables de entorno, pero la página de códigos no lo es, COBOL para Linux utiliza la página de códigos del sistema predeterminada para la combinación de idioma y código de país.

Pueden aplicarse varias páginas de códigos a la combinación de idioma y código de país. Si no desea que el sistema Linux seleccione una página de códigos predeterminada, debe especificar la página de códigos explícitamente.

UTF-8: La codificación UTF-8 está soportada para cualquier combinación de idioma y código de país.

Tareas relacionadas
"Especificación de la página de códigos para datos de caracteres" en la página 214
"Utilización de variables de entorno para especificar un entorno local" en la página 215
"Acceso al entorno local activo y a los valores de página de códigos" en la página 223
"Establecimiento de variables de entorno" en la página 229

Referencias relacionadas
"Entornos locales y páginas de códigos soportadas" en la página 217
"Variables de entorno de compilador y tiempo de ejecución" en la página 230

Tipos de mensajes para los que hay traducciones disponibles

Los mensajes siguientes están habilitados para el soporte de idioma nacional: compilador, mensajes de interfaz de usuario de tiempo de ejecución y depuradory cabeceras de listado (incluidos los formatos de fecha y hora basados en el entorno local).

Se utilizan el texto y los formatos adecuados, tal como se especifica en el entorno local activo, para estos mensajes y las cabeceras de listado.

Consulte la referencia relacionada a continuación para obtener información sobre las variables de entorno LANG y NLSPATH, que afectan al idioma y al entorno local de los mensajes.

Conceptos relacionados
"El entorno local activo" en la página 213

Tareas relacionadas
"Utilización de variables de entorno para especificar un entorno local" en la página 215

Referencias relacionadas
"Variables de entorno de compilador y tiempo de ejecución" en la página 230

Entornos locales y páginas de códigos soportadas

La tabla siguiente muestra los entornos locales que podrían estar disponibles en el sistema y las páginas de códigos que están soportadas para cada entorno local. COBOL para Linux da soporte a los entornos locales que están disponibles en el sistema en tiempo de compilación y en tiempo de ejecución.

Para consultar los entornos locales disponibles en el sistema, entre lo siguiente:

```
locale -a
```

Tabla 20. ***Entornos locales y páginas de códigos soportados***

Nombre de entorno local[1]	**Idioma[2]**	**País o región[3]**	**Páginas de códigos basadas en ASCII[4]**	**Páginas de códigos EBCDIC[5]**	**Grupo de idiomas**
any			utf-8	Las páginas de códigos EBCDIC que son aplicables al entorno local se basan en las partes *language* y *COUNTRY* del nombre de entorno local independientemente del valor de página de códigos del entorno local.	
ar_AE	Árabe	Emiratos Árabes Unidos	iso88596	IBM-16804, IBM-420	Árabe
be_BY	Bielorruso	Belarús	iso88595	IBM-1025, IBM-1154	Latín 5
bg_BG	Búlgaro	Bulgaria	iso88595	IBM-1025, IBM-1154	Latín 5
ca_ES	Catalán	España	iso88591	IBM-285, IBM-1145	Latín 1
cs_CZ	Checo	República Checa	iso88592	IBM-870, IBM-1153	Latín 2
da_DK	Danés	Dinamarca	iso88591	IBM-277, IBM-1142	Latín 1
de_CH	Alemán	Suiza	iso88591	IBM-500, IBM-1148	Latín 1
de_DE	Alemán	Alemania	iso88591	IBM-273, IBM-1141	Latín 1
el_GR	Griego	Grecia	iso88597	IBM-4971, IBM-875	Griego
en_AU	Español	Australia	iso88591	IBM-037, IBM-1140	Latín 1
en_GB	Español	Reino Unido	iso88591	IBM-037, IBM-1140	Latín 1
en_US	Español	Estados Unidos	iso88591	IBM-037, IBM-1140	Latín 1
es_ES	Español	Sudáfrica	iso88591	IBM-037, IBM-1140	Latín 1
es_ES	Español	España	iso88591	IBM-284, IBM-1145	Latín 1
fi_FI	Finés	Finlandia	iso88591	IBM-278, IBM-1143	Latín 1

Tabla 20. ***Entornos locales y páginas de códigos soportados*** *(continuación)*

Nombre de entorno local[1]	Idioma[2]	País o región[3]	Páginas de códigos basadas en ASCII[4]	Páginas de códigos EBCDIC[5]	Grupo de idiomas
fr_BE	Francés	Bélgica	iso88591	IBM-297, IBM-1148	Latín 1
fr_CA	Francés	Canadá	iso88591	IBM-037, IBM-1140	Latín 1
fr_CH	Francés	Suiza	iso88591	IBM-500, IBM-1148	Latín 1
fr_FR	Francés	Francia	iso88591	IBM-297, IBM-1148	Latín 1
hr_HR	Croata	Croacia	iso88592	IBM-870, IBM-1153	Latín 2
hu_HU	Húngaro	Hungría	iso88592	IBM-870, IBM-1153	Latín 2
is_IS	Islandés	Islandia	iso88591	IBM-871, IBM-1149	Latín 1
it_CH	Italiano	Suiza	iso88591	IBM-500, IBM-1148	Latín 1
it_IT	Italiano	Italia	iso88591	IBM-280, IBM-1144	Latín 1
iw_IL	Hebreo	Israel	iso88598	IBM-12712, IBM-424	Hebreo
ja_JP	Japonés	Japón	IBMeucjp	IBM-930, IBM-939, IBM-1390, IBM-1399	Idiomas ideográficos
ko_KR	Coreano	Corea, República de	euckr	IBM-933, IBM-1364	Idiomas ideográficos
lt_LT	Lituano	Lituania	IBMiso885913	n/d	Lituano
lv_LV	Letón	Letonia	IBMiso885913	n/d	Letón
mk_MK	Macedonio	Macedonia	iso88595	IBM-1025, IBM-1154	Latín 5
nl_BE	Holandés	Bélgica	iso8859-1	IBM-500, IBM-1148	Latín 1
nl_NL	Holandés	Países Bajos	iso88591	IBM-037, IBM-1140	Latín 1
no_NO	Noruego	Noruega	iso88591	IBM-277, IBM-1142	Latín 1
pl_PL	Polaco	Polonia	iso88592	IBM-870, IBM-1153	Latín 2
pt_BR	Portugués	Brasil	iso88591	IBM-037, IBM-1140	Latín 1
pt_PT	Portugués	Alemania	iso88591	IBM-037, IBM-1140	Latín 1
ro_RO	Rumano	Rumanía	iso88592	IBM-870, IBM-1153	Latín 2
ru_RU	Ruso	Federación Rusa	iso88595	IBM-1025, IBM-1154	Latín 5
sk_SK	Eslovaco	Eslovaquia	iso88592	IBM-870, IBM-1153	Latín 2
sl_SI	Esloveno	Eslovenia	iso8859-2	IBM-870, IBM-1153	Latín 2
sq_AL	Albanés	Albania	iso88591	IBM-500, IBM-1148	Latín 1
ser_SE	Sueco	Suecia	iso88591	IBM-278, IBM-1143	Latín 1
th_TH	Tailandés	Tailandia	tis620	IBM-9030	Tailandés
tr_TR	Turco	Turquía	iso88599	IBM-1026, IBM-1155	Turco
uk_UA	Ucraniano	Ucrania	iso88595	IBM-1123, IBM-1154	Latín 5

Tabla 20. ***Entornos locales y páginas de códigos soportados*** *(continuación)*

Nombre de entorno local[1]	Idioma[2]	País o región[3]	Páginas de códigos basadas en ASCII[4]	Páginas de códigos EBCDIC[5]	Grupo de idiomas
CI_zh_CN	Chino	China	gb18030	IBM-1388	Idiomas ideográficos
zh_TW	Chino (tradicional)	Taiwán	IBMeuctw	IBM-1371, IBM-937	Idiomas ideográficos

1. Muestra las combinaciones válidas de código de idioma ISO y código de país ISO (*language_COUNTRY*) que están soportadas. Las mayúsculas y minúsculas de cada carácter del nombre de entorno local que se muestra en la tabla son significativas y es posible que no reflejen las mayúsculas y minúsculas de un nombre de entorno local con una página de códigos específica seleccionada (o implícita) para el entorno local. Consulte los resultados del mandato "locale -a" para ver las mayúsculas y minúsculas adecuadas de cada carácter para el nombre de entorno local seleccionado.
2. Muestra el idioma asociado.
3. Muestra el país o región asociado.
4. Muestra las páginas de códigos que son válidas como el ID de página de códigos para el entorno local que tiene el valor *language_COUNTRY* correspondiente. Estas entradas de tabla no son definitivas. Para obtener la lista actual de entornos locales válidos, consulte la documentación del sistema para ver la versión y configuración específicas de Linux que está ejecutando. El entorno local que seleccione debe ser válido, es decir, debe estar instalado tanto donde desarrolle como donde ejecute el programa.
5. Muestra las páginas de códigos que son válidas como el ID de página de códigos para el entorno local que tiene el valor *language_COUNTRY* correspondiente. Estas páginas de códigos son válidas como contenido para la variable de entorno EBCDIC_CODEPAGE. Si la variable de entorno EBCDIC_CODEPAGE no está establecida, la entrada de página de códigos más a la derecha que se muestra en esta columna se selecciona como página de códigos EBCDIC para el entorno local correspondiente.

Tareas relacionadas
“Especificación de la página de códigos para datos de caracteres” en la página 214
“Utilización de variables de entorno para especificar un entorno local” en la página 215

Control de la secuencia de clasificación con un entorno local

Varias operaciones, como comparaciones, ordenación y fusión, utilizan la secuencia de clasificación que está en vigor para el programa y los elementos de datos. Cómo se controla la secuencia de clasificación depende de la página de códigos en vigor para la clase de los datos: alfabético, alfanumérico, DBCS o nacional.

Acerca de esta tarea

Una secuencia de clasificación basada en el entorno local para elementos que son alfabéticos de clase, alfanuméricos o DBCS sólo se aplica cuando la opción de compilador `COLLSEQ(LOCALE)` está en vigor, no cuando `COLLSEQ(BIN)` o `COLLSEQ(EBCDIC)` está en vigor. De forma similar, una secuencia de clasificación basada en el entorno local para elementos nacionales de clase sólo se aplica cuando la opción de compilador `NCOLLSEQ(LOCALE)` está en vigor, no cuando `NCOLLSEQ(BIN)` está en vigor.

Si la opción de compilador `COLLSEQ(LOCALE)` o `NCOLLSEQ(LOCALE)` está en vigor, el entorno local de tiempo de compilación se utiliza para los elementos de lenguaje que tienen sintaxis o reglas semánticas que se ven afectadas por el orden de clasificación basado en el entorno local, como por ejemplo:

- Frase THRU en una cláusula de nombre de condición VALUE
- Frase *literal-3* THRU *literal-4* en la sentencia EVALUATE

- *literal-1* THRU *literal-2* frase en la cláusula ALPHABET
- Posiciones ordinales de caracteres especificados en la cláusula SYMBOLIC CHARACTERS
- Frase THRU en la cláusula CLASS

Si la opción de compilador COLLSEQ(LOCALE) está en vigor, el orden de clasificación de las claves alfanuméricas en las sentencias SORT y MERGE siempre se basa en el entorno local de ejecución.

Tareas relacionadas
"Especificación del orden de clasificación" en la página 6
"Establecimiento de criterios de ordenación o fusión" en la página 169
"Especificación de la página de códigos para datos de caracteres" en la página 214
"Utilización de variables de entorno para especificar un entorno local" en la página 215
"Control de la secuencia de clasificación alfanumérica con un entorno local" en la página 220
"Control de la secuencia de clasificación DBCS con un entorno local" en la página 221
"Control de la secuencia de clasificación nacional con un entorno local" en la página 222
"Acceso al entorno local activo y a los valores de página de códigos" en la página 223

Referencias relacionadas
"Entornos locales y páginas de códigos soportadas" en la página 217
"COLLSEQ" en la página 275
"NCOLLSEQ" en la página 290

Control de la secuencia de clasificación alfanumérica con un entorno local

La secuencia de clasificación para los caracteres alfanuméricos de un solo byte para la secuencia de clasificación del programa se basa en el entorno local en tiempo de compilación o en el entorno local en tiempo de ejecución.

Acerca de esta tarea

Si especifica PROGRAM COLLATING SEQUENCE en el programa fuente, el orden de clasificación se establece en tiempo de compilación y se utiliza independientemente del entorno local en tiempo de ejecución. Si, en su lugar, establece la secuencia de clasificación utilizando la opción de compilador COLLSEQ , el entorno local en tiempo de ejecución tiene prioridad.

Si la página de códigos en vigor es una página de códigos ASCII de un solo byte, puede especificar las cláusulas siguientes en el párrafo SPECIAL-NAMES :

- cláusula ALPHABET
- cláusula SYMBOLIC CHARACTERS
- cláusula CLASS

Si especifica estas cláusulas cuando la página de códigos fuente en vigor incluye caracteres DBCS, las cláusulas se diagnosticarán y tratarán como comentarios. Las reglas del nombre de alfabeto COBOL definido por el usuario y los caracteres simbólicos asumen una secuencia de clasificación de carácter por carácter, no una secuencia de clasificación que depende de una secuencia de varios caracteres.

Si especifica la cláusula PROGRAM COLLATING SEQUENCE en el párrafo OBJECT-COMPUTER , el orden de clasificación que está asociado con el *nombre-alfabeto* se utiliza para determinar el valor de verdad de las comparaciones alfanuméricas. La cláusula PROGRAM COLLATING SEQUENCE también se aplica a las claves de clasificación y fusión de USAGE DISPLAY a menos que especifique la frase COLLATING SEQUENCE en la sentencia SORT o MERGE .

Si no especifica la frase COLLATING SEQUENCE o la cláusula PROGRAM COLLATING SEQUENCE , el orden de clasificación en vigor es NATIVE de forma predeterminada y se basa en el valor de entorno local activo. Este valor se aplica a las sentencias SORT y MERGE y al orden de clasificación del programa.

La secuencia de clasificación afecta al proceso de los elementos siguientes:

- Cláusula ALPHABET (por ejemplo, *literal-1* THRU *literal-2*)

- Especificaciones de SYMBOLIC CHARACTERS
- Especificaciones de rango de VALUE para elementos level-88 , condiciones de relación y sentencias SORT y MERGE

Tareas relacionadas
“Especificación del orden de clasificación” en la página 6
“Control de la secuencia de clasificación con un entorno local” en la página 219
“Control de la secuencia de clasificación DBCS con un entorno local” en la página 221
“Control de la secuencia de clasificación nacional con un entorno local” en la página 222
“Establecimiento de criterios de ordenación o fusión” en la página 169

Referencias relacionadas
“COLLSEQ” en la página 275
Clases y categorías de datos (*COBOL for Linux en x86 Consulta de lenguaje*)
Comparaciones alfanuméricas (*COBOL for Linux en x86 Consulta de lenguaje*)

Control de la secuencia de clasificación DBCS con un entorno local

La secuencia de clasificación basada en el entorno local en tiempo de ejecución siempre se aplica a los datos DBCS, excepto para comparaciones de literales.

Acerca de esta tarea

Puede utilizar un elemento de datos o literal de clase DBCS en una condición de relación con cualquier operador relacional. El otro operando de debe ser de clase DBCS o de clase nacional, o ser un grupo alfanumérico. No se realiza ninguna distinción entre elementos DBCS y elementos DBCS editados.

Cuando compara dos operandos DBCS, la secuencia de clasificación la determina el entorno local activo si la opción de compilador COLLSEQ(LOCALE) está en vigor. De lo contrario, la secuencia de clasificación viene determinada por los valores binarios de los caracteres DBCS. La cláusula PROGRAM COLLATING SEQUENCE no tiene ningún efecto en las comparaciones que implican elementos de datos o literales de clase DBCS.

Cuando compara un elemento DBCS con un elemento nacional, el operando DBCS se trata como si se trasladara a un elemento nacional elemental de la misma longitud que el operando DBCS. Los caracteres DBCS se convierten a representación nacional y la comparación continúa con dos operandos de caracteres nacionales.

Cuando compara un elemento DBCS con un grupo alfanumérico de , no se realiza ninguna conversión o edición. La comparación continúa como para dos operandos de caracteres alfanuméricos. La comparación opera en bytes de datos sin tener en cuenta la representación de datos.

Tareas relacionadas
“Especificación del orden de clasificación” en la página 6
“Utilización de literales DBCS” en la página 211
“Control de la secuencia de clasificación con un entorno local” en la página 219
“Control de la secuencia de clasificación alfanumérica con un entorno local” en la página 220
“Control de la secuencia de clasificación nacional con un entorno local” en la página 222

Referencias relacionadas
“COLLSEQ” en la página 275
Clases y categorías de datos (*COBOL for Linux en x86 Consulta de lenguaje*)
Comparaciones alfanuméricas (*COBOL for Linux en x86 Consulta de lenguaje*)
Comparaciones DBCS (*COBOL for Linux en x86 Consulta de lenguaje*)
Comparaciones de grupo (*COBOL for Linux en x86 Consulta de lenguaje*)

Control de la secuencia de clasificación nacional con un entorno local

Puede utilizar literales nacionales o elementos de datos de USAGE NATIONAL en una condición de relación con cualquier operador relacional. La cláusula PROGRAM COLLATING SEQUENCE no tiene ningún efecto en las comparaciones que implican operandos nacionales.

Acerca de esta tarea

Utilice la opción de compilador NCOLLSEQ(LOCALE) para efectuar comparaciones basadas en el algoritmo para el orden de clasificación que está asociado con el entorno local activo en tiempo de ejecución. Si NCOLLSEQ(BINARY) está en vigor, la secuencia de clasificación viene determinada por los valores binarios de los caracteres nacionales.

Las claves utilizadas en una sentencia SORT o MERGE pueden ser class national sólo si la opción NCOLLSEQ(BIN) está en vigor.

Tareas relacionadas
"Comparación de datos nacionales (UTF-16)" en la página 205
"Control de la secuencia de clasificación con un entorno local" en la página 219
"Control de la secuencia de clasificación DBCS con un entorno local" en la página 221
"Establecimiento de criterios de ordenación o fusión" en la página 169

Referencias relacionadas
"NCOLLSEQ" en la página 290
Clases y categorías de datos (*COBOL for Linux en x86 Consulta de lenguaje*)
Comparaciones nacionales (*COBOL for Linux en x86 Consulta de lenguaje*)

Funciones intrínsecas que dependen de la secuencia de clasificación

Las siguientes funciones intrínsecas dependen de las posiciones ordinales de los caracteres.

Para una página de códigos ASCII, estas funciones intrínsecas están soportadas en función del orden de clasificación en vigor. Para una página de códigos EUC o una página de códigos que incluye caracteres DBCS, se asume que las posiciones ordinales de caracteres de un solo byte corresponden a las representaciones hexadecimales de los caracteres de un solo byte. Por ejemplo, la posición ordinal para 'A' es 66 (X'41' + 1) y la posición ordinal para '*' es 43 (X'2A' + 1).

Tabla 21. ***Funciones intrínsecas que dependen del orden de clasificación***

Función intrínseca	Devuelve:	Comentarios
CHAR	Carácter que corresponde al argumento de posición ordinal	
MAX	Contenido del argumento que contiene el valor máximo	Los argumentos pueden ser alfabéticos, alfanuméricos, nacionales o numéricos.[1]
MIN	Contenido del argumento que contiene el valor mínimo	Los argumentos pueden ser alfabéticos, alfanuméricos, nacionales o numéricos.[1]
ORD	Posición ordinal del argumento de carácter	
ORD-MAX	Posición ordinal entera en la lista de argumentos del argumento que contiene el valor máximo	Los argumentos pueden ser alfabéticos, alfanuméricos, nacionales o numéricos.[1]
ORD-MIN	Posición ordinal entera en la lista de argumentos del argumento que contiene el valor mínimo	Los argumentos pueden ser alfabéticos, alfanuméricos, nacionales o numéricos.[1]

Tabla 21. **Funciones intrínsecas que dependen del orden de clasificación** *(continuación)*		
Función intrínseca	**Devuelve:**	**Comentarios**
1. La página de códigos y la secuencia de clasificación no son aplicables cuando la función tiene argumentos numéricos.		

Estas funciones intrínsecas no están soportadas para el tipo de datos DBCS.

Tareas relacionadas
“Especificación del orden de clasificación” en la página 6
“Comparación de datos nacionales (UTF-16)” en la página 205
“Control de la secuencia de clasificación con un entorno local” en la página 219

Acceso al entorno local activo y a los valores de página de códigos

Para verificar el entorno local que está en vigor en el momento de la compilación, compruebe las últimas líneas del listado del compilador.

Acerca de esta tarea

Para algunas aplicaciones, es posible que desee verificar el entorno local y la página de códigos EBCDIC que están activas en tiempo de ejecución y convertir un ID de página de códigos al CCSID correspondiente. Puede utilizar rutinas de biblioteca invocables para realizar estas consultas y conversiones.

Para acceder al entorno local y a la página de códigos EBCDIC que están activos en tiempo de ejecución, llame a la función de biblioteca _iwzGetLocaleCP de la siguiente manera:

```
CALL "_iwzGetLocaleCP" USING output1, output2
```

La variable *output1* es un elemento alfanumérico de 20 caracteres que representa el valor de entorno local terminado en nulo en el formato siguiente:

- Código de idioma de dos caracteres
- Un subrayado (_)
- Código de país de dos caracteres
- Un punto (.)
- El valor de página de códigos para el entorno local

Por ejemplo, en_US.IBM-1252 es el valor de entorno local del código de idioma en, el código de país USy la página de códigos IBM-1252.

La variable *output2* es un elemento alfanumérico de 10 caracteres que representa el ID de página de códigos EBCDIC terminado en nulo en vigor, como IBM-1140.

Para convertir un ID de página de códigos al CCSID correspondiente, llame a la función de biblioteca _iwzGetCCSID de la siguiente manera:

```
CALL "_iwzGetCCSID" USING input, output RETURNING returncode
```

input es un elemento alfanumérico que representa un ID de página de códigos terminado en nulo.

La*salida* es un elemento binario de 4 bytes con signo, como por ejemplo uno definido como PIC S9(5) COMP-5. Se devuelve el CCSID que corresponde a la serie de ID de página de códigos de entrada o el código de error -1.

returncode es un elemento de datos binarios de 4 bytes con signo, que se establece de la forma siguiente:

0
Satisfactorio.

1
El ID de página de códigos es válido pero no tiene un CCSID asociado; *salida* se establece en -1.

-1
El ID de página de códigos no es una página de códigos válida; la *salida* se establece en -1.

Para llamar a estos servicios, debe utilizar el convenio de interfaz de llamada de `PGMNAME(MIXED)` y `NODYNAM` .

"Ejemplo: obtener y convertir un ID de página de códigos" en la página 224

Tareas relacionadas
Capítulo 11, "Establecimiento del entorno local", en la página 213

Referencias relacionadas
"DYNAM" en la página 281
"PGMNAME" en la página 292
Capítulo 14, "Sentencias de direccionamiento de compilador", en la página 309

Ejemplo: obtener y convertir un ID de página de códigos

El ejemplo siguiente muestra cómo puede utilizar los servicios invocables _iwzGetLocaleCP y _iwzGetCCSID para recuperar el entorno local y la página de códigos EBCDIC que están en vigor, respectivamente, y convertir un ID de página de códigos al CCSID correspondiente.

```
cbl pgmname(lm)
       Identification Division.
       Program-ID.  "Samp1".
       Data Division.
       Working-Storage Section.
       01 locale-in-effect.
          05 ll-cc             pic x(5).
          05 filler-period     pic x.
          05 ASCII-CP          Pic x(14).
       01 EBCDIC-CP            pic x(10).
       01 CCSID                pic s9(5) comp-5.
       01 RC                   pic s9(5) comp-5.
       01 n                    pic 99.

       Procedure Division.
       Get-locale-and-codepages section.
       Get-locale.
           Display "Start Samp1."
           Call "_iwzGetLocaleCP"
             using locale-in-effect, EBCDIC-CP
           Move 0 to n
           Inspect locale-in-effect
             tallying n for characters before initial x'00'
           Display "locale in effect: " locale-in-effect (1 : n)
           Move 0 to n
          Inspect EBCDIC-CP
             tallying n for characters before initial x'00'
          Display "EBCDIC code page in effect: "
            EBCDIC-CP (1 : n).

      Get-CCSID-for-EBCDIC-CP.
          Call "_iwzGetCCSID" using EBCDIC-CP, CCSID returning RC
          Evaluate RC
            When 0
              Display "CCSID for " EBCDIC-CP (1 : n) " is " CCSID
            When 1
              Display EBCDIC-CP (1 : n)
                      " does not have a CCSID value."
            When other
              Display EBCDIC-CP (1 : n) " is not a valid code page."
          End-Evaluate.

      Done.
          Goback.
```

Si establece el entorno local en ja_JP.IBM-943 (`set LC_ALL=ja_JP.IBM-943`), la salida del programa de ejemplo es:

```
Start Samp1.
locale in effect: ja_JP.IBM-943
EBCDIC code page in effect: IBM-1399
CCSID for IBM-1399 is 0000001399
```

Tareas relacionadas

“Utilización de variables de entorno para especificar un entorno local” en la página 215

Parte 3. Compilación, enlace, ejecución y depuración del programa

Capítulo 12. Compilación, enlace y ejecución de programas

En las secciones siguientes se explica cómo establecer variables de entorno, compilar, enlazar, ejecutar, y corregir errores.

Acerca de esta tarea

Tareas relacionadas
“Establecimiento de variables de entorno” en la página 229
“Compilación de programas” en la página 239
“Corrección de errores en el programa fuente” en la página 245
“Enlace de programas” en la página 250
“Corrección de errores en el enlace” en la página 253
“ejecutar programas” en la página 254

Referencias relacionadas
“Opciones cob2” en la página 248

Establecimiento de variables de entorno

Las variables de entorno se utilizan para establecer los valores que necesitan los programas. Especifique el valor de una variable de entorno utilizando el mandato `export` o la función putenv () POSIX . Si no establece una variable de entorno, se aplica un valor predeterminado o la variable no está definida.

Acerca de esta tarea

Una *variable de entorno* define algún aspecto de un entorno de usuario o un entorno de programa que puede variar. Por ejemplo, puede utilizar la variable de entorno COBPATH para definir las ubicaciones en las que el tiempo de ejecución COBOL puede encontrar un programa cuando otro programa lo llama dinámicamente. Las variables de entorno las utilizan las bibliotecas de compilador y de tiempo de ejecución.

Al instalar IBM COBOL for Linux en x86, el proceso de instalación establece variables de entorno para acceder al compilador de COBOL para Linux y a las bibliotecas de tiempo de ejecución. Para compilar y ejecutar un programa COBOL simple, las únicas variables de entorno que se debe establecer es LANG, y sólo se debe establecer si desea utilizar mensajes que no sean los mensajes en_US predeterminados.

Puede cambiar el valor de una variable de entorno en cualquiera de los dos lugares utilizando el mandato `export` :

- En el indicador de un shell de mandatos (por ejemplo, en una ventana XTERM). Esta definición de variable de entorno se aplica a los programas (procesos o procesos hijo) que se ejecutan desde dicho shell o desde cualquiera de sus descendientes (es decir, cualquier shell llamado directa o indirectamente desde dicho shell).
- En el archivo .profile del directorio de inicio. Si define variables de entorno en el archivo .profile, los valores de estas variables se definen automáticamente siempre que inicia la sesión de a Linux y los valores se aplican a todos los procesos de shell.

También puede establecer variables de entorno desde un programa COBOL utilizando la función putenv () POSIX y acceder a las variables de entorno utilizando la función POSIX de getenv ().

Algunas variables de entorno (como COBPATH y NLSPATH) definen directorios en los que buscar archivos. Si se listan varias vías de acceso de directorio, están delimitadas por dos puntos. Las vías de acceso definidas por variables de entorno se evalúan en orden, desde la primera vía de acceso hasta la última del

mandato `export` . Si se definen varios archivos que tienen el mismo nombre en las vías de acceso de una variable de entorno, se utiliza la primera copia localizada del archivo.

Por ejemplo, el siguiente mandato `export` establece la variable de entorno COBPATH (que define las ubicaciones en las que el tiempo de ejecución COBOL puede encontrar programas accedidos dinámicamente) para incluir dos directorios, el primero de los cuales se busca primero:

```
export COBPATH=/users/me/bin:/mytools/bin
```

"Ejemplo: establecimiento y acceso a variables de entorno" en la página 238

Tareas relacionadas
"Adaptación de la compilación" en la página 243

Referencias relacionadas
"Variables de entorno de compilador y tiempo de ejecución" en la página 230
"Variables de entorno de compilador" en la página 233
"Variables de entorno de ejecución" en la página 234

Variables de entorno de compilador y tiempo de ejecución

COBOL para Linux utiliza las variables de entorno siguientes que son comunes al compilador y al tiempo de ejecución.

DB2DBDFT
Especifica la base de datos que se debe utilizar para los programas que contienen sentencias de SQL incorporado o que utilizan el sistema de archivos Db2.

DBCS_CODEPAGE
Especifica una página de códigos DBCS aplicable a los datos DBCS, incluidos los literales DBCS y los elementos de datos DBCS.

Para establecer la página de códigos DBCS, emita el mandato siguiente, donde *página de códigos* es el nombre de una página de códigos DBCS soportada por las bibliotecas de conversión ICU (International Components for Unicode), por ejemplo, IBM-943 o IBM-EUCjp:

```
export DBCS_CODEPAGE=codepage
```

Si DBCS_CODEPAGE no está establecido, se utiliza la página de códigos DBCS predeterminada asociada con el entorno local actual.

LANG
Especifica el entorno local (tal como se describe en la tarea relacionada sobre el uso de variables de entorno para especificar un entorno local). LANG también influye en el valor de la variable de entorno NLSPATH tal como se describe a continuación.

Por ejemplo, el mandato siguiente establece el nombre de entorno local de idioma en EE.UU. Inglés:

```
export LANG=en_US
```

LC_ALL
Especifica el entorno local. Un valor de entorno local que utiliza LC_ALL altera temporalmente cualquier valor que utilice LANG o cualquier otra variable de entorno LC_*xx* (tal como se describe en la tarea relacionada sobre el uso de variables de entorno para especificar un entorno local).

LC_COLLATE
Especifica el comportamiento de ordenación para el entorno local. Este valor se altera temporalmente si se especifica LC_ALL.

LC_CTYPE
Especifica la página de códigos para el entorno local. Este valor se altera temporalmente si se especifica LC_ALL.

LC_MESSAGES
Especifica el idioma de los mensajes para el entorno local. Este valor se altera temporalmente si se especifica LC_ALL.

HORA_LC
Determina el entorno local para la información de formato de fecha y hora. Este valor se altera temporalmente si se especifica LC_ALL.

vía_acceso_LD_LIBRARY_PATH
Especifica las vías de acceso de directorio que deben utilizarse para las bibliotecas compartidas y los programas de salida de compilador definidos por el usuario que la opción de compilador EXIT ha identificado.

NLSPATH
Especifica el nombre completo de vía de acceso de los catálogos de mensajes y los archivos de ayuda y utiliza el formato *directory_name*/%L/%N, donde %L se sustituye por el valor especificado por la variable de entorno LANG. %N se sustituye por el nombre del catálogo de mensajes.

COBOL para Linux instala el catálogo de mensajes del compilador en `/opt/ibm/cobol/1.2.0/usr/share/locale/`*xx*, y el catálogo de mensajes de tiempo de ejecución en `/opt/ibm/cobol/rte/usr/share/locale/`*xx* en donde *xx* es cualquier idioma al que COBOL para Linux da soporte. El valor predeterminado es en_US.

Cuando establezca NLSPATH, asegúrese de añadir a NLSPATH en lugar de sustituirlo. Otros programas pueden utilizar esta variable de entorno. Por ejemplo:

```
DIR=xxxx
NLSPATH=$DIR/%L/%N:$NLSPATH
export NLSPATH
```

xxxx es el directorio donde se ha instalado COBOL. El directorio *xxxx* debe contener un directorio *xxxx*/en_US (en el caso de un EE.UU. Configuración del idioma inglés) que contiene el catálogo de mensajes COBOL.

Los mensajes en los idiomas siguientes se incluyen con el producto:

en_US
Español

ja_JP
Japonés

Puede especificar los idiomas para los mensajes y para el valor de entorno local de forma diferente. Por ejemplo, puede establecer la variable de entorno LANG en en_US y establecer la variable de entorno LC_ALL en `ja_JP.eucjp`. En este ejemplo, cualquier compilador COBOL o mensaje de tiempo de ejecución estará en inglés, mientras que los datos ASCII nativos (`DISPLAY` o `DISPLAY-1`) del programa se tratan como codificados en la página de códigos ja_JP.eucjp (página de códigos EUC japonesa).

El compilador utiliza la combinación de los valores de la variable de entorno NLSPATH y LANG para acceder al catálogo de mensajes. Si NLSPATH se ha establecido de forma válida pero LANG no se ha establecido en uno de los valores de entorno local mostrados anteriormente, se genera un mensaje de aviso y el compilador toma el valor predeterminado del catálogo de mensajes en_US. Si el valor NLSPATH no es válido, se genera un mensaje de error de terminación.

La biblioteca de tiempo de ejecución también utiliza NLSPATH para acceder al catálogo de mensajes. Si NLSPATH no se ha establecido correctamente, los mensajes de tiempo de ejecución aparecen en una forma abreviada. El compilador y el tiempo de ejecución gestionan automáticamente NLSPATH, por lo que no es necesario que maneje NLSPATH usted mismo.

TMPDIR
Especifica la ubicación de los archivos de trabajo temporales utilizados por el compilador y el tiempo de ejecución. Si este valor no está establecido, el valor predeterminado es el directorio actual.

Por ejemplo:

```
export TMPDIR=/tmp
```

TZ

Describe la información de huso horario que debe utilizar el entorno local. TZ tiene el formato siguiente:

```
export TZ=SSS[+|-]nDDD[,sm,sw,sd,st,em,ew,ed,et,shift]
```

Si TZ no está presente, el valor predeterminado es EST5EDT, el valor de entorno local predeterminado. Si sólo se especifica el huso horario estándar, el valor predeterminado de *n* (diferencia en horas respecto a GMT) es 0 en lugar de 5.

Si proporciona valores para cualquiera de *sm*, *sw*, *sd*, *st*, *em*, *ew*, *ed*, *et*o *shift*, debe proporcionar valores para todos ellos. Si alguno de estos valores no es válido, la sentencia completa se considera no válida y la información de huso horario no se cambia.

Por ejemplo, la sentencia siguiente establece el huso horario estándar en CST, establece el horario de verano en CDT y establece una diferencia de seis horas entre CST y UTC. No establece ningún valor para el inicio y el final del horario de verano.

```
export TZ=CST6CDT
```

Otros valores posibles son PST8PDT para Estados Unidos del Pacífico y MST7MDT para Estados Unidos de montaña.

Tareas relacionadas

“Utilización de variables de entorno para especificar un entorno local” en la página 215

Referencias relacionadas

“Entornos locales y páginas de códigos soportadas” en la página 217
“Variables de parámetro de entorno TZ” en la página 232

Variables de parámetro de entorno TZ

Los valores de la variable TZ se definen a continuación.

Tabla 22. ***Variables de parámetro de entorno deTZ***

VARIABLE	Descripción	Valor predeterminado
SSS	Identificador de huso horario estándar. Debe tener tres caracteres, debe empezar por una letra y puede contener espacios.	EST
n	Diferencia (en horas) entre el huso horario estándar y la hora universal coordinada (UTC), anteriormente denominada hora media de Greenwich (GMT). Un número positivo indica zonas horarias al oeste del meridiano de Greenwich. Un número negativo indica los husos horarios al este del meridiano de Greenwich.	5
DDD	Identificador de zona de horario de verano (DST). Debe tener tres caracteres, debe empezar por una letra y puede contener espacios.	EDT
sm	Mes de inicio (1 a 12) de DST	4
sw	Semana de inicio (-4 a 4) de DST	1
sd	Día de inicio de DST: de 0 a 6 si *sw* no es cero; de 1 a 31 si *sw* es cero	0

Tabla 22. ***Variables de parámetro de entorno deTZ*** *(continuación)*		
VARIABLE	**Descripción**	**Valor predeterminado**
st	Hora de inicio (en segundos) de DST	3600
em	Mes final (1 a 12) de DST	10
nuevo	Semana final (-4 a 4) de DST	-1
ed	Día final de DST: de 0 a 6 si *ew* no es cero; de 1 a 31 si *ew* es cero	0
et	Hora de finalización (en segundos) de DST	7200
mayús	Cantidad de cambio de tiempo (en segundos)	3600

Variables de entorno de compilador

COBOL para Linux utiliza varias variables de entorno de sólo compilador, tal como se muestra a continuación.

Puesto que las palabras COBOL no distinguen entre mayúsculas y minúsculas, todas las letras de las palabras COBOL se tratan como mayúsculas, incluyendo *nombre-biblioteca* y *nombre-texto*. Por lo tanto, los nombres de variables de entorno que corresponden a dichos nombres deben estar en mayúsculas. Por ejemplo, el nombre de variable de entorno que corresponde a `COPY MyCopy` es MYCOPY.

COBCPYEXT
Especifica los sufijos de archivo que se deben utilizar en las búsquedas de libros de copias cuando la sentencia `COPY name` no indica un sufijo de archivo. Especifique uno o más sufijos de archivo de con o sin puntos iniciales. Separe varios sufijos de archivo con un espacio o una coma.

Si no se ha definido COBCPYEXT, se buscan los sufijos siguientes: CPY, CBL, COB y los equivalentes en minúsculas cpy, cbl y cob.

DIR_MANDATO
Especifica el directorio en el que se graba el archivo de listado del compilador. Especifique cualquier vía de acceso válida. Para indicar una vía de acceso absoluta, especifique una barra inclinada inicial. De lo contrario, la vía de acceso es relativa al directorio actual. Una barra inclinada final es opcional.

Si no se ha definido COBLSTDIR, el listado del compilador se graba en el directorio actual.

COBOPT
Especifica las opciones del compilador. Para especificar varias opciones de compilador, separe cada opción con un espacio o una coma. Escriba la lista de opciones entre comillas si la lista contiene espacios en blanco o caracteres que son significativos para el shell de mandatos. Por ejemplo:

```
export COBOPT="TRUNC(OPT) TERMINAL"
```

Los valores predeterminados se aplican a las opciones de compilador individuales.

Nota: El compilador interpreta determinados caracteres de script de shell como se indica a continuación:

- Un signo igual (=) se interpreta como un paréntesis izquierdo, (
- Un signo de dos puntos (:) se interpreta como un paréntesis derecho,)
- Un subrayado (_) se interpreta como una comilla simple (')

Puede añadir un carácter de escape de barra inclinada invertida (\) para evitar la interpretación y, por lo tanto, para pasar caracteres en las series. Si desea que la barra inclinada invertida (\) se represente a sí misma (en lugar de como un carácter de escape), utilice la barra inclinada invertida doble (\\).

nombre-biblioteca
Si especifica *nombre-biblioteca* como palabra definida por el usuario, el nombre se utiliza como variable de entorno y el valor de la variable de entorno se utiliza como vía de acceso en la que localizar el libro de copias.

Si no especifica un nombre de biblioteca, el compilador busca las vías de acceso de biblioteca en el orden siguiente:

1. Directorio actual
2. Vías de acceso especificadas por la opción `-I`*xxx* , si se ha establecido
3. Vías de acceso especificadas por la variable de entorno SYSLIB

La búsqueda finaliza cuando se encuentra el archivo.

Para obtener más detalles, consulte la documentación de la sentencia `COPY` en la referencia relacionada sobre las sentencias de dirección de compilador.

SYSLIB
Especifica las vías de acceso que deben utilizarse para las sentencias COBOL `COPY` que tienen nombres de texto que no están calificados por los nombres de biblioteca. También especifica las vías de acceso que se van a utilizar para las sentencias `SQL INCLUDE` .

nombre-texto
Si especifica *text-name* como una palabra definida por el usuario, el nombre se utiliza como una variable de entorno, y el valor de la variable de entorno se utiliza como el nombre de archivo y posiblemente el nombre de vía de acceso del libro de copias.

Para especificar varios nombres de vía de acceso, delícelos con dos puntos (`:`).

Para obtener más detalles, consulte la documentación de la sentencia `COPY` en la referencia relacionada sobre las sentencias de dirección de compilador.

Conceptos relacionados
“CoprocesadorDb2” en la página 401

Tareas relacionadas
“Utilización de SQL INCLUDE con el coprocesador de Db2” en la página 402

Referencias relacionadas
“Opciones cob2” en la página 248
“opciones de compilador” en la página 265
Capítulo 14, “Sentencias de direccionamiento de compilador”, en la página 309

Variables de entorno de ejecución

La biblioteca de tiempo de ejecución COBOL utiliza las siguientes variables de entorno de sólo tiempo de ejecución.

nombre-asignación
La palabra definida por el usuario que especifique (en la cláusula `ASSIGN`) para el nombre de archivo externo para un archivo COBOL; por ejemplo, `OUTPUTFILE` en la siguiente cláusula `ASSIGN` :

```
SELECT CARPOOL ASSIGN TO OUTPUTFILE
```

En tiempo de ejecución, establezca la variable de entorno en el nombre del archivo del sistema que desea asociar con el archivo COBOL. Por ejemplo:

```
export OUTPUTFILE=january.car_results
```

Después de emitir el mandato anterior, las sentencias de entrada/salida para el archivo COBOL `CARPOOL` operan en el archivo del sistema `january.car_results` en el directorio actual.

Si no establece la variable de entorno, o la establece en la serie vacía, COBOL utiliza el nombre literal de la variable de entorno como nombre de archivo del sistema (OUTPUTFILE en el directorio actual para el ejemplo anterior de ASSIGN).

La cláusula ASSIGN puede especificar un archivo almacenado en un sistema de archivos que no sea el predeterminado, como por ejemplo el sistema de archivos de lenguaje estándar (STL) o el sistema de archivos delimitado secuencial de registros (RSD). Por ejemplo:

```
SELECT CARPOOL ASSIGN TO STL-OUTPUTFILE
```

En este caso, sigue definiendo la variable de entorno OUTPUTFILE (no STL-OUTPUTFILE).

CICS_TK_SFS_SERVER
Especifica el nombre de servidor SFS CICS completo. Por ejemplo:

```
export CICS_TK_SFS_SERVER=/.:/cics/sfs/sfsServer
```

COBOL_BCD_NOVALIDATE
Especifica si el elemento de datos decimal empaquetado se valida antes de las operaciones de conversión y aritméticas.

Cuando esta variable de entorno se establece en 1, se omite la validación de dígito y signo realizada antes de la conversión y las operaciones aritméticas.

El uso de esta variable de entorno no afecta a la prueba de clase numérica que prueba si un elemento de datos IS NUMERIC o IS NOT NUMERIC.

COBPATH
Especifica las vías de acceso de directorio que debe utilizar el tiempo de ejecución COBOL para localizar programas a los que se accede dinámicamente como, por ejemplo, bibliotecas compartidas. COBPATH se busca primero (y si no se establece, el valor predeterminado es el directorio actual (" ./")), seguido de LD_LIBRARY_PATH.

Por ejemplo:

```
export COBPATH=/pgmpath/pgmshlib
```

COBRTOPT
Especifica las opciones de tiempo de ejecución COBOL.

Separe las opciones de tiempo de ejecución con una coma o dos puntos. Utilice paréntesis o signos de igual (=) como delimitador para las subopciones. Las opciones no distinguen entre mayúsculas y minúsculas. Por ejemplo, los dos mandatos siguientes son equivalentes:

```
export COBRTOPT="CHECK(ON):UPSI(00000000)"
export COBRTOPT=check=on,upsi=00000000
```

Si especifica más de un valor para una opción de tiempo de ejecución determinada, prevalecerá el valor situado más a la derecha.

Se aplican los valores predeterminados para las opciones de tiempo de ejecución individuales. Para obtener detalles, consulte la referencia relacionada sobre las opciones de tiempo de ejecución.

EBCDIC_CODEPAGE
Especifica una página de códigos EBCDIC aplicable a los datos EBCDIC procesados por programas compilados con la opción de compilador CHAR(EBCDIC) o CHAR(S390) .

Para establecer la página de códigos EBCDIC, emita el mandato siguiente, donde *página de códigos* es el nombre de la página de códigos que se va a utilizar:

```
export EBCDIC_CODEPAGE=codepage
```

Si EBCDIC_CODEPAGE no está establecido, la página de códigos EBCDIC predeterminada se selecciona basándose en el entorno local actual, tal como se describe en la referencia relacionada sobre los entornos locales y las páginas de códigos soportadas. Cuando la opción de compilador `CHAR(EBCDIC)` está en vigor y hay varias páginas de códigos EBCDIC aplicables al entorno local en vigor, debe establecer la variable de entorno EBCDIC_CODEPAGE a menos que la página de códigos EBCDIC predeterminada para el entorno local sea aceptable.

CICS_SFS_DATA_VOLUME
Especifica el nombre del volumen de datos SFS en el que se van a crear los archivos SFS. Por ejemplo:

```
export CICS_SFS_DATA_VOLUME=sfs_SFS_SERV
```

Este volumen de datos debe haberse definido en el servidor SFS al que accede la aplicación.

Si no se establece esta variable, se utiliza el nombre predeterminado sfs_SSFS_SERVER .

CICS_SFS_INDEX_VOLUME
Especifica el nombre del volumen de datos SFS en el que se van a crear los archivos de índice alternativos. Por ejemplo:

```
export CICS_SFS_INDEX_VOLUME=sfs_SFS_SERV
```

Este volumen de datos debe haberse definido en el servidor SFS al que accede la aplicación.

Si esta variable no está establecida, los archivos de índice alternativos se crean en el mismo volumen de datos que los archivos de índice base correspondientes.

CICS_VSAM_AUTO_FLUSH
Especifica si todos los cambios en los archivos SFS de CICS para cada operación de entrada-salida se confirman en disco antes de que se devuelva el control a la aplicación (es decir, si la característica de *fuerza operativa* de SFS está habilitada). Puede mejorar el rendimiento de las aplicaciones que utilizan archivos SFS especificando 0FF para esta variable de entorno. Por ejemplo:

```
export CICS_VSAM_AUTO_FLUSH=OFF
```

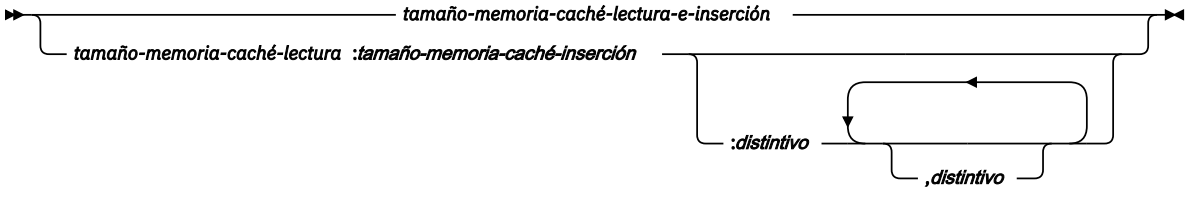
Cuando esta variable de entorno se establece en 0FF, SFS utiliza una estrategia *lazy-write* ; es decir, es posible que los cambios en los archivos SFS no se confirmen en el disco hasta que se cierren los archivos.

Si el almacenamiento en memoria caché del lado del cliente de SFS está en vigor (es decir, la variable de entorno CICS_VSAM_CACHE se establece en un valor distinto de cero válido), el valor de CICS_VSAM_AUTO_FLUSH se ignora. La fuerza operativa está inhabilitada.

Si el almacenamiento en memoria caché del lado del cliente de SFS no está en vigor, el valor de CICS_VSAM_AUTO_FLUSH toma el valor predeterminado 0N.

CICS_VSAM_CACHE
Especifica si el almacenamiento en memoria caché del lado del cliente está habilitado para archivos SFS. Puede mejorar el rendimiento de las aplicaciones que utilizan archivos SFS habilitando el almacenamiento en memoria caché del lado del cliente.

Sintaxis de CICS_VSAM_CACHE



Las unidades de tamaño están en números de páginas. Un tamaño de cero indica que el almacenamiento en memoria caché está inhabilitado. Los distintivos posibles son:

ALLOW_DIRTY_READS
Elimina la restricción para el almacenamiento en memoria caché de lectura de que los archivos a los que se accede deben estar bloqueados.

INSERTS_DESPITE_UNIQUE_INDICES
Elimina la restricción para el almacenamiento en memoria caché de inserción que las inserciones sólo se almacenan en la memoria caché si todos los índices activos para archivos en clúster y todos los índices alternativos activos para archivos relativos y secuenciados de entrada permiten duplicados.

Por ejemplo, el mandato siguiente establece el tamaño de la memoria caché de lectura en 16 páginas y el tamaño de la memoria caché de inserción en 64 páginas. También habilita las lecturas sucias y habilita las inserciones a pesar de los índices exclusivos:

```
export CICS_VSAM_CACHE=16:64:ALLOW_DIRTY_READS, \
    INSERTS_DESPITE_UNIQUE_INDICES
```

De forma predeterminada, el almacenamiento en memoria caché del lado del cliente no está habilitado.

CICS_SFS_CACHE_ <*nombre_archivo*>
Especifica si el almacenamiento en memoria caché del lado del cliente está habilitado para un archivo SFS específico, donde puede sustituir <*nombre_archivo*> por el nombre de archivo SFS. Si el nombre de archivo contiene (.) caracteres, debe sustituirlos por (_) (subrayado). Este valor ayudaría a habilitar la memoria caché del lado del cliente basándose en la operación de archivo (lectura o escritura) realizada en cualquier archivo específico. Puede especificar CICS_SFS_CACHE_ <*nombre_archivo*> utilizando la misma especificación de sintaxis que el valor CICS_SFS_CACHE.

Por ejemplo, para habilitar el almacenamiento en memoria caché del lado del cliente específico del archivo para un archivo con el nombre VSAM.FILE1.TESTFILE, que siempre se utiliza para la operación de grabación SFS, el entorno sólo se puede especificar para habilitar la memoria caché de operación WRITE:

```
export CICS_SFS_CACHE_FILE1_TESTFILE=0:512
```

CICS_SFS_RDM_CACHE
Especifica si el almacenamiento en memoria caché del lado del cliente está inhabilitado para los archivos SFS utilizados para la operación de lectura aleatoria. Especifique el valor de entorno como 0 para inhabilitar la memoria caché de operación de lectura del lado del cliente para todos los archivos que se abren para la operación de lectura aleatoria:

```
export CICS_SFS_RDM_CACHE=0
```

El valor CICS_SFS_CACHE es un valor global que es aplicable a todos los archivos SFS, y no se recomienda establecer la memoria caché de operación de lectura para los archivos que se abren para operaciones de lectura aleatoria. El valor CICS_VSAM_RDM_CACHE se puede utilizar para inhabilitar la memoria caché de operaciones de lectura del lado del cliente de SFS.

CICS_SFS_PREALLOC_ <*nombre_archivo*>

Especifica el número de páginas preasignadas para un archivo, cuando el programa crea el archivo por primera vez en el servidor SFS. Puede sustituir <*nombre_archivo*> por el nombre de archivo SFS. Si el nombre de archivo contiene (.) caracteres, debe sustituirlos por (_) (subrayado). Por ejemplo, para crear un archivo SFS denominado TESTFILE y preasignar 2000 páginas, establezca la siguiente variable de entorno:

```
export CICS_SFS_PREALLOC_TESTFILE=2000
```

COBCORE
Especifica que se colocará la ubicación de los archivos principales generados por el tiempo de ejecución. Si no se especifica, el valor predeterminado es el directorio de trabajo actual. Si el nombre termina con un carácter de porcentaje (%), se crea un nombre de archivo con el nombre de programa y la indicación de fecha y hora.

COBOUTDIR
Especifica un directorio donde se crean todos los archivos creados por sentencias COBOL DISPLAY (SYSOUT, CONSOLE, SYSPUNCH) y los archivos principales (COBCORE), si la variable no incluye una vía de acceso.

PATH
Especifica las vías de acceso de directorio de los programas ejecutables.

SYSIN, SYSIPT, SYSOUT, SYSLIST, SYSLST, CONSOLE, SYSPUNCH, SYSPCH
Estos nombres de entorno COBOL se utilizan como los nombres de variable de entorno que corresponden a los nombres nemotécnicos utilizados en las sentencias ACCEPT y DISPLAY .

Si la variable de entorno no está establecida, de forma predeterminada SYSIN y SYSIPT se dirigen al dispositivo de entrada lógico (teclado). SYSOUT, SYSLIST, SYSLST y CONSOLE se dirigen al dispositivo de salida lógica del sistema (pantalla). SYSPUNCH y SYSPCH no tienen asignado un valor predeterminado y no son válidos a menos que los defina explícitamente. Si el valor de cualquiera de estas variables termina con un carácter de porcentaje (%), se crea un nombre de archivo con el nombre de programa y la indicación de fecha y hora.

Por ejemplo, el mandato siguiente define CONSOLE:

```
export CONSOLE=/users/mypath/myfile
```

A continuación, se podría utilizar CONSOLE junto con el siguiente código fuente:

```
SPECIAL-NAMES.
    CONSOLE IS terminal
    . . .
    DISPLAY 'Hello World' UPON terminal
```

BASEDATOS_MONGOBD_VF

Especifica la base de datos que se debe utilizar para los programas que utilizan el sistema de archivos MongoDB .

URI VFS_MONGODB_URI

Especifica el URI de conexión que identifica la ubicación del servidor MongoDB .

Tareas relacionadas
“Identificación de archivos” en la página 117
“Utilización de archivos SFS” en la página 156
“Utilización de archivos MongoDB” en la página 154
“Mejora del rendimiento de SFS” en la página 159

Referencias relacionadas
“char” en la página 272
Capítulo 15, “Opciones de tiempo de ejecución”, en la página 315

Ejemplo: establecimiento y acceso a variables de entorno

El ejemplo siguiente muestra cómo puede acceder y establecer variables de entorno desde un programa COBOL llamando a las funciones POSIX estándar getenv () y putenv ().

Puesto que getenv () y putenv () son funciones C, debe pasar los argumentos BY VALUE. Pase series de caracteres como punteros BY VALUE que apunten a series terminadas en nulo. Compile programas que llamen a estas funciones con las opciones NODYNAM y PGMNAME(LONGMIXED) .

```
 CBL pgmname(longmixed),nodynam
 Identification division.
 Program-id. "envdemo".
 Data division.
 Working-storage section.
 01 P pointer.
 01 PATH pic x(5) value Z"PATH".
 01 var-ptr pointer.
 01 var-len pic 9(4) binary.
 01 putenv-arg pic x(14) value Z"MYVAR=ABCDEFG".
 01 rc pic 9(9) binary.
 Linkage section.
 01 var pic x(5000).
 Procedure division.
* Retrieve and display the PATH environment variable
     Set P to address of PATH
     Call "getenv" using by value P returning var-ptr
     If var-ptr = null then
         Display "PATH not set"
     Else
         Set address of var to var-ptr
         Move 0 to var-len
         Inspect var tallying var-len
           for characters before initial X"00"
         Display "PATH = " var(1:var-len)
     End-if
* Set environment variable MYVAR to ABCDEFG
     Set P to address of putenv-arg
     Call "putenv" using by value P returning rc
     If rc not = 0 then
         Display "putenv failed"
         Stop run
     End-if
     Goback.
```

Compilación de programas

Puede compilar los programas COBOL desde la línea de mandatos o utilizando un script de shell o un archivo make.

Acerca de esta tarea

Si tiene un origen COBOL que no es deIBM o de formato libre, es posible que primero tenga que utilizar el programa de utilidad de conversión de origen, scu, para ayudar a convertir el origen para que se pueda compilar. Para ver un resumen de las funciones de scu , escriba el mandato scu -h. Para obtener más detalles, consulte la página man para scu, o consulte la referencia relacionada sobre el programa de utilidad de conversión de origen de .

Especificación de opciones de compilador: existen varias formas en las que puede especificar las opciones que se deben utilizar al compilar programas COBOL. Por ejemplo, puede:

- Establezca la variable de entorno COBOPT desde la línea de mandatos.
- Especifique opciones de compilador como opciones para el mandato cob2 . Puede utilizar este mandato en la línea de mandatos, en un script de shell o en un archivo make.
- Utilice sentencias PROCESS (CBL) o ∗CONTROL . Una opción que especifique utilizando PROCESS altera temporalmente cualquier otra especificación de opción.

Para obtener más detalles sobre cómo establecer las opciones de compilador y la prioridad relativa de los métodos para establecerlas, consulte la referencia relacionada sobre las opciones de compilador en conflicto.

Tareas relacionadas
“Compilación desde la línea de mandatos” en la página 240

“Compilación utilizando archivos de proceso por lotes o archivos de mandatos de scripts de shell” en la página 241
“Especificación de opciones de compilador en la sentencia PROCESS (CBL)” en la página 242
“Modificación de la configuración de compilador predeterminada” en la página 242
Capítulo 21, “Portabilidad de aplicaciones entre plataformas y compiladores COBOL”, en la página 455

Referencias relacionadas
“Variables de entorno de compilador” en la página 233
“Opciones cob2” en la página 248
“opciones de compilador” en la página 265
“Opciones de compilador en conflicto” en la página 267

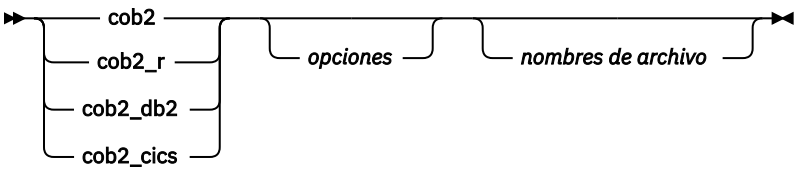
"Migración de Enterprise COBOL for z/OS a COBOL for Linux en x86" en la *Guía de migración*
Formato de referencia (*COBOL for Linux en x86 Consulta de lenguaje*)
Programa de utilidad de conversión de origen (scu) (*COBOL for Linux en x86 Consulta de lenguaje*)

Compilación desde la línea de mandatos

Para compilar un programa COBOL desde la línea de mandatos, emita el mandato cob2. Este mandato también invoca el enlazador.

Acerca de esta tarea

Sintaxis del mandato cob2



Para compilar varios archivos, especifique los nombres de archivo en cualquier posición de la línea de mandatos, utilizando espacios para separar opciones y nombres de archivo. Por ejemplo, los dos mandatos siguientes son equivalentes:

```
cob2 -g filea.cbl fileb.cbl -q"flag(w)"
cob2 filea.cbl -g -q"flag(w)" fileb.cbl
```

El mandato cob2 acepta opciones de compilador y enlazador en cualquier orden en la línea de mandatos. Las opciones que especifique se aplicarán a todos los archivos de la línea de mandatos.

Sólo se pasan al compilador los archivos de origen que tienen el sufijo .cbl o .cob . Todos los demás archivos se pasan al enlazador.

La ubicación predeterminada para la entrada y salida del compilador es el directorio actual.

El mandato cob2 es seguro para hebras. `cob2_r` se proporciona para la compatibilidad con COBOL for AIX, pero utiliza las mismas opciones predeterminadas que el mandato cob2 . Se recomiendan `cob2_db2` y `cob2_cics` al programar para un entorno Db2 o CICS. Cuando se utilizan junto con las stanzas correspondientes en el archivo `cob2.cfg` , estos mandatos utilizan las opciones de compilador, las vías de acceso de instalación y las vías de acceso de biblioteca de tiempo de ejecución necesarias para dichos entornos. Consulte “Modificación de la configuración de compilador predeterminada” en la página 242 para obtener más información sobre las opciones predeterminadas.

Nota: El compilador interpreta determinados caracteres de script de shell como se indica a continuación:

- Un signo igual (=) se interpreta como un paréntesis izquierdo, (
- Un signo de dos puntos (:) se interpreta como un paréntesis derecho,)

- Un subrayado (_) se interpreta como una comilla simple (')

Puede añadir un carácter de escape de barra inclinada invertida (\) para evitar la interpretación y, por lo tanto, para pasar caracteres en las series. Si desea que la barra inclinada invertida (\) se represente a sí misma (en lugar de como un carácter de escape), utilice la barra inclinada invertida doble (\\).

"Ejemplos: utilización de cob2 para compilar" en la página 241

Tareas relacionadas
"Modificación de la configuración de compilador predeterminada" en la página 242
"Enlace de programas" en la página 250

Referencias relacionadas
"Opciones cob2" en la página 248
"opciones de compilador" en la página 265

Ejemplos: utilización de cob2 para compilar

Los ejemplos siguientes muestran la salida producida para varias invocaciones de `cob2` .

Tabla 23. ***Salida del mandato cob2***

Para compilar:	**Especifique:**	**Se generan estos archivos:**
alfa.cbl	`cob2 -c alpha.cbl`	alpha.o
alpha.cbl y beta.cbl	`cob2 -c alpha.cbl beta.cbl`	alpha.o, beta.o
alpha.cbl con las opciones `LIST` y `ADATA`	`cob2 -qlist,adata alpha.cbl`	alpha.wlist, alpha.o[1], alpha.lst, alpha.adt y a.out

1. Si el enlace es satisfactorio, este archivo se suprime.

"Ejemplos: utilización de cob2 para enlazar" en la página 251

Tareas relacionadas
"Compilación desde la línea de mandatos" en la página 240

Referencias relacionadas
"DATOS" en la página 268
"lista" en la página 287

Compilación utilizando archivos de proceso por lotes o archivos de mandatos de scripts de shell

Puede utilizar un script de shell para automatizar el mandato `cob2`.

Acerca de esta tarea

Sin embargo, para evitar que se pase una sintaxis no válida al mandato, siga estas directrices:

- Utilice un signo de igual y dos puntos en lugar de paréntesis para delimitar las subopciones del compilador.
- Utilice caracteres de subrayado en lugar de comillas simples para delimitar las subopciones del compilador.
- No utilice espacios en blanco en la serie de opción a menos que ponga la serie entre comillas (" ").

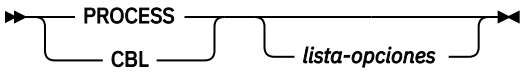
Tabla 24. ***Ejemplos de sintaxis de opción de compilador en un script de shell***	
Utilizar en script de shell	**Utilizar en línea de mandatos**
`-qFLOAT=NATIVE:,CHAR=NATIVE:`	`-qFLOAT(NATIVE),CHAR(NATIVE)`
`-qEXIT=INEXIT=_String_,MYMODULE::`	`-qEXIT(INEXIT('String',MYMODULE))`

Especificación de opciones de compilador en la sentencia PROCESS (CBL)

Dentro de un programa COBOL, puede codificar la mayoría de opciones de compilador en sentencias PROCESS (CBL). Codifique las sentencias antes de la cabecera `IDENTIFICATION DIVISION` y antes de cualquier línea de comentario o sentencia de dirección de compilador.

Acerca de esta tarea

Sintaxis de la sentencia PROCESS (CBL)



Si no utiliza un campo de secuencia, puede iniciar una sentencia PROCESS en la columna 1 o después. Si utiliza un campo de secuencia, el número de secuencia debe empezar en la columna 1 y debe contener seis caracteres; el primer carácter debe ser numérico. Si se utiliza con un campo de secuencia, PROCESS puede empezar en la columna 8 o después.

Puede utilizar CBL como sinónimo de PROCESS. De la misma forma, CBL puede empezar en la columna 1 o después si no utiliza un campo de secuencia. Si se utiliza con un campo de secuencia, CBL puede empezar en la columna 8 o después.

Debe finalizar las sentencias PROCESS y CBL en o antes de la columna 72 si el programa utiliza el formato fuente fijo, o la columna 252 si el programa utiliza el formato fuente ampliado.

Utilice uno o más espacios en blanco para separar una sentencia PROCESS o CBL de la primera opción de *options-list*. Separe las opciones con una coma o un espacio en blanco. No inserte espacios entre opciones individuales y sus subopciones.

Puede codificar más de una sentencia PROCESS o CBL . Si lo hace, las declaraciones deben ir una detrás de la otra sin declaraciones intervinientes. No puede continuar las opciones entre varias sentencias PROCESS o CBL .

Referencias relacionadas
“FORMATOORIGEN” en la página 298
Formato de referencia (*COBOL for Linux en x86 Consulta de lenguaje*)
Declaración CBL (PROCESS) (*COBOL for Linux en x86 Consulta de lenguaje*)

Modificación de la configuración de compilador predeterminada

Las opciones predeterminadas utilizadas por el mandato `cob2` se obtienen del archivo de configuración, que es de forma predeterminada `/opt/ibm/cobol/1.2.0/etc/cob2.cfg`. Puede visualizar las opciones utilizadas por `cob2` especificando la opción `-#` en el mandato.

Acerca de esta tarea

Si está utilizando el archivo de configuración predeterminado, el mandato `cob2 -# abc.cbl` muestra una salida similar a la siguiente:

```
exec: /opt/ibm/cobol/1.2.0/usr/bin/cob3_64 -qADDR(64) hello.cbl
exec: /usr/bin/gcc -m64 -shared -fPIC -rdynamic -fasynchronous-unwind-tables
-Wl,--hash-style=gnu -Wl,--export-dynam -Wl,-Bsymbolic -Wl,--build-id
```

```
-Wl,--enable-new-dtags -Wl,-zrelro -Wl,-znow -Wl,-zdefs -Wl,-z,noexecstack
-Wl,-znotext -pie -fPIE -fwhole-program -Wl,--as-needed -Wl,--no-allow-shlib-undefined
-Wl,--push-state -Wl,--allow-shlib-undefined -Wl,--pop-state -L/opt/ibm/cobol/1.2.0/usr/lib/
-L/opt/ibm/cobol/rte/usr/lib/ -lcob2_64s -lcob2_64r
-Wl,-rpath,/opt/ibm/cobol/rte/usr/lib/:/opt/ibm/cobol/rte/:
/opt/ibm/cobol/1.2.0/usr/lib/:/opt/ibm/cobol/rte/usr/lib/:/opt/ibm/cics/lib
```

Puede modificar el `cob2.cfg` archivo de configuración para cambiar las opciones predeterminadas.

En lugar de modificar el archivo de configuración predeterminado, puede adaptar una copia del archivo para sus propósitos.

Nota: El compilador interpreta determinados caracteres de script de shell como se indica a continuación:

- Un signo igual (=) se interpreta como un paréntesis izquierdo, (
- Un signo de dos puntos (:) se interpreta como un paréntesis derecho,)
- Un subrayado (_) se interpreta como una comilla simple (')

Puede añadir un carácter de escape de barra inclinada invertida (\) para evitar la interpretación y, por lo tanto, para pasar caracteres en las series. Si desea que la barra inclinada invertida (\) se represente a sí misma (en lugar de como un carácter de escape), utilice la barra inclinada invertida doble (\\).

Tareas relacionadas
“Adaptación de la compilación” en la página 243

Referencias relacionadas
“Opciones cob2” en la página 248
“Stanzas en el archivo de configuración” en la página 244

Adaptación de la compilación

Para adaptar una compilación a sus necesidades, puede cambiar una copia del archivo de configuración predeterminado y utilizarlo de varias maneras.

Acerca de esta tarea

El archivo de configuración `/opt/ibm/cobol/1.2.0/etc/cob2.cfg` tiene secciones, denominadas *stanzas*, que empiezan por `cob2:`. Para listar estas stanzas, utilice el mandato `ls /opt/ibm/cobol/1.2.0/usr/bin/cob2*`. La lista resultante muestra estas líneas:

```
/opt/ibm/cobol/1.2.0/usr/bin/cob2
/opt/ibm/cobol/1.2.0/usr/bin/cob2_r
/opt/ibm/cobol/1.2.0/usr/bin/cob2_cics
/opt/ibm/cobol/1.2.0/usr/bin/cob2_db2
/opt/ibm/cobol/1.2.0/usr/bin/cob2_oracle
```

Los mandatos `cob2` y `cob2_r` ejecutan el mismo módulo; sin embargo, `cob2` utiliza la stanza `cob2` y `cob2_r` utiliza la stanza `cob2_r` .

Puede adaptar la compilación realizando los pasos siguientes:

Procedimiento

1. Haga una copia del archivo de configuración predeterminado `/opt/ibm/cobol/1.2.0/etc/cob2.cfg`.
2. Cambie la copia para dar soporte a requisitos de compilación específicos u otros entornos de compilación COBOL.
3. (Opcional) Emita el mandato `cob2` con la opción `-#` para visualizar el efecto de los cambios.
4. Utilice la copia en lugar de `/opt/ibm/cobol/1.2.0/etc/cob2.cfg`.

Resultados

Para utilizar la copia de `cob2.cfg` en cada invocación del compilador, cambie el enlace simbólico en el directorio `etc.d` con ese nombre para que apunte a la copia. El compilador lee automáticamente el archivo de configuración al que apunta este enlace simbólico.

Para utilizar de forma selectiva una copia modificada del archivo de configuración para una compilación específica, emita el mandato `cob2` con la opción `-F` . Por ejemplo, para utilizar `/u/myhome/myconfig.cfg` en lugar de `/opt/ibm/cobol/1.2.0/etc/cob2.cfg` como archivo de configuración para compilar `myfile.cbl`, emita este mandato:

```
cob2 myfile.cbl -F/u/myhome/myconfig.cfg
```

Si añade su propia stanza, como `mycob2`, puede especificarla con la opción `-F` :

```
cob2 myfile.cbl -F/u/myhome/myconfig.cfg:mycob2
```

O puede definir un mandato `mycob2` :

```
ln -s /usr/bin/cob2 /u/myhome/mycob2
mycob2 myfile.cbl -F/u/myhome/myconfig
```

Cualquier directorio que nombre en el mandato `ln` (por ejemplo, `/u/myhome` anterior) debe estar en la variable de entorno PATH.

Para obtener una lista de los atributos de cada stanza, consulte la referencia relacionada sobre las stanzas.

Tareas relacionadas
"Establecimiento de variables de entorno" en la página 229

Referencias relacionadas
"Opciones cob2" en la página 248
"Stanzas en el archivo de configuración" en la página 244

Stanzas en el archivo de configuración

Una stanza del archivo de configuración puede contener cualquiera de varios atributos, tal como se muestra en la tabla siguiente.

Tabla 25. ***Atributos de stanza***

Atributo	**Descripción**
COMPOPTS	Serie de opciones de compilador, separadas por comas o espacios. Preceda cada opción por un distintivo `-q` , o preceda a toda la serie, entre comillas, por un distintivo `-q` . Si algún valor de opción contiene una coma, dicha opción debe estar entre comillas.
coprocesador, coprocessor_64	Vía de acceso a la biblioteca de coprocesador CICS o Db2 .
runlib2, runlib2_64	Bibliotecas de tiempo de ejecución adicionales en las que enlazar al trabajar con CICS, Db2u Oracle.
utilizar	Stanza de la que se toman los atributos, además de la stanza local. Para atributos de valor único, los valores de la stanza `use` se aplican si no se proporciona ningún valor en la stanza local o predeterminada. Para listas separadas por comas, los valores de la stanza `use` se añaden a los valores de la stanza local.

Tareas relacionadas
“Modificación de la configuración de compilador predeterminada” en la página 242
“Adaptación de la compilación” en la página 243

Referencias relacionadas
“Opciones cob2” en la página 248

Corrección de errores en el programa fuente

Los mensajes sobre errores de código fuente indican dónde se ha producido el error (`LINEID`). El texto de un mensaje le indica cuál es el problema. Con esta información, puede corregir el programa fuente.

Acerca de esta tarea

Aunque debe intentar corregir los errores, no siempre es necesario corregir el código fuente para cada mensaje de diagnóstico. Puede dejar un mensaje de nivel de aviso o de nivel informativo en un programa sin mucho riesgo, y puede decidir que la recodificación y la compilación necesarias para eliminar el mensaje no valgan la pena. Sin embargo, los errores de nivel grave y de nivel de error indican una probable anomalía del programa y deben corregirse.

En contraste con los cuatro niveles inferiores de gravedad, un error irrecuperable (nivel U) podría no ser el resultado de un error en el programa fuente. Podría provenir de una falla en el propio compilador o en el sistema operativo. En tales casos, el problema debe resolverse, porque el compilador se fuerza a finalizar antes y no produce un código de objeto completo o un listado completo. Si el mensaje aparece para un programa que tiene muchos errores de sintaxis de nivel S, corrija esos errores y vuelva a compilar el programa. También puede resolver problemas de configuración de trabajos (como la falta de definiciones de archivo o almacenamiento insuficiente para el proceso del compilador) realizando cambios en el trabajo de compilación. Si la configuración del trabajo de compilación es correcta y ha corregido los errores de sintaxis de nivel S, debe ponerse en contacto con IBM para investigar otros errores de nivel U.

Después de corregir los errores en el programa fuente, vuelva a compilar el programa. Si esta segunda compilación es satisfactoria, continúe con el paso de edición de enlaces. Si el compilador sigue encontrando problemas, repita el procedimiento anterior hasta que sólo se devuelvan mensajes informativos.

Tareas relacionadas
“Generación de una lista de mensajes del compilador” en la página 246
“Enlace de programas” en la página 250

Referencias relacionadas
“Mensajes y listados de errores detectados por el compilador” en la página 246

Códigos de gravedad para mensajes de diagnóstico del compilador

Las condiciones que el compilador puede detectar se encuentran en cinco niveles o categorías de gravedad.

Tabla 26. ***Códigos de gravedad para mensajes de diagnóstico del compilador***

Nivel o categoría de mensaje	**Código de retorno**	**objetivo**
Informativo (I)	0	Para informarle. No es necesaria ninguna acción y el programa se ejecuta correctamente.
Aviso (W)	4	Para indicar un posible error. El programa probablemente se ejecuta correctamente tal como está escrito.

*Tabla 26. **Códigos de gravedad para mensajes de diagnóstico del compilador** (continuación)*		
Nivel o categoría de mensaje	**Código de retorno**	**objetivo**
Error (E)	8	Para indicar una condición que es definitivamente un error. El compilador ha intentado corregir el error, pero es posible que los resultados de la ejecución del programa no sean los que esperaba. Debe corregir el error.
Grave (S)	12	Indicar una condición que es un error grave. El compilador no ha podido corregir el error. El programa no se ejecuta correctamente y no se debe intentar la ejecución. Es posible que no se haya creado el código de objeto.
Irrecuperable (U)	16	Para indicar una condición de error de tal magnitud que la compilación se ha terminado.

El código de retorno final al final de la compilación es generalmente el código de retorno más alto que se ha producido para cualquier mensaje durante la compilación.

Sin embargo, puede suprimir los mensajes de diagnóstico del compilador o cambiar sus gravedades, lo que puede tener un efecto sobre el código de retorno de compilación final. Para obtener detalles, consulte la información relacionada.

Tareas relacionadas
"Personalización de gravedades de mensajes del compilador" en la página 617

Referencias relacionadas
"Proceso de MSGEXIT" en la página 616

Generación de una lista de mensajes del compilador

Puede generar un listado completo de mensajes de diagnóstico del compilador con sus números de mensaje, gravedades y texto compilando un programa que tenga nombre de programa ERRMSG.

Acerca de esta tarea

Puede codificar sólo el párrafo `PROGRAM-ID` , como se muestra a continuación, y omitir el resto del programa.

```
Identification Division.
Program-ID. ErrMsg.
```

Tareas relacionadas
"Personalización de gravedades de mensajes del compilador" en la página 617

Referencias relacionadas
"Mensajes y listados de errores detectados por el compilador" en la página 246
"Formato de los mensajes de diagnóstico del compilador" en la página 247

Mensajes y listados de errores detectados por el compilador

A medida que el compilador procesa el programa fuente, comprueba si hay errores de lenguaje COBOL y emite mensajes de diagnóstico. Estos mensajes se clasifican en el listado del compilador (sujeto a la opción `FLAG`).

El archivo de listado del compilador tiene el mismo nombre que el archivo de origen del compilador, pero con la extensión sufijo .lst. Por ejemplo, el archivo de listado para myfile.cbl es myfile.lst. El archivo de listado se graba en el directorio desde el que se ha emitido el mandato `cob2` .

Cada mensaje del listado proporciona información sobre la naturaleza del problema, su gravedad y la fase del compilador que lo ha detectado. Siempre que sea posible, el mensaje proporciona instrucciones específicas para corregir un error.

Los mensajes para los errores encontrados durante el proceso de las opciones del compilador, las sentencias CBL y PROCESS y las sentencias BASIS, COPYo REPLACE se visualizan cerca de la parte superior del listado.

Los mensajes para errores de compilación (ordenados por número de línea) se visualizan cerca del final del listado para cada programa.

Un resumen de todos los problemas encontrados durante la compilación se muestra cerca de la parte inferior del listado.

Tareas relacionadas
“Corrección de errores en el programa fuente” en la página 245
“Generación de una lista de mensajes del compilador” en la página 246

Referencias relacionadas
“Formato de los mensajes de diagnóstico del compilador” en la página 247
“Códigos de gravedad para mensajes de diagnóstico del compilador” en la página 245
“BANDERA” en la página 284

Formato de los mensajes de diagnóstico del compilador

Cada mensaje emitido por el compilador tiene un número de línea fuente, un identificador de mensaje y texto de mensaje.

Cada mensaje tiene el formato siguiente:

```
nnnnnn IGYppxxxx-l message-text
```

nnnnnn
Número de la sentencia fuente de la última línea que el compilador estaba procesando. Los números de sentencia de origen se listan en la salida impresa de origen del programa. Si ha especificado la opción NUMBER en tiempo de compilación, los números son los números de programa fuente originales. Si ha especificado NONUMBER, los números son los generados por el compilador.

IGY
Prefijo que identifica que el compilador COBOL ha emitido el mensaje.

pp
Dos caracteres que identifican qué fase o subfase del compilador ha detectado la condición que ha dado como resultado un mensaje. Como programador de aplicaciones, puede ignorar esta información. Si está diagnosticando un error de compilador sospechoso, póngase en contacto con IBM para obtener soporte.

xxxx
Número de cuatro dígitos que identifica el mensaje.

l
Carácter que indica el nivel de gravedad del mensaje: I, W, E, S o U.

texto-mensaje
El texto del mensaje; para un mensaje de error, una breve explicación de la condición que ha causado el error.

Consejo: Si ha utilizado la opción FLAG para suprimir mensajes, es posible que haya errores adicionales en el programa.

Referencias relacionadas
“Códigos de gravedad para mensajes de diagnóstico del compilador” en la página 245
“BANDERA” en la página 284

Opciones cob2

Las opciones listadas a continuación se aplican a el mandato de invocación `cob2`.

Opciones que se aplican a la compilación

-c
Compila programas pero no los enlaza.

-comprc_ok=*n*
Controla el comportamiento tras la devolución del compilador. Si el código de retorno es menor o igual que *n*, el mandato continúa en el paso de enlace, o en el caso de sólo compilación, sale con un código de retorno cero. Si el código de retorno generado por el compilador es mayor que *n*, el mandato sale con el mismo código de retorno devuelto por el compilador.

El valor predeterminado es `-comprc_ok=4`.

-host
Establece estas opciones de compilador para la representación de datos de Enterprise COBOL for z/OS y la semántica del lenguaje:

- `CHAR(EBCDIC)`
- `COLLSEQ(EBCDIC)`
- `NCOLLSEQ(BIN)`
- `FLOAT(S390)`

La opción `-host` cambia el formato de los argumentos de línea de mandatos del programa COBOL de una matriz de punteros a una serie de caracteres EBCDIC que tiene un prefijo de media palabra que contiene la longitud de la serie. Para obtener información adicional, consulte la tarea relacionada siguiente sobre el uso de argumentos de línea de mandatos.

Si utiliza la opción `-host` y desea que una rutina principal en un archivo de objeto C sea el punto de entrada principal de la aplicación, debe utilizar la opción de enlazador `-cmain` tal como se describe a continuación.

-I*xxx*
Añade la vía de acceso *xxx* a los directorios en los que se buscarán los libros de copias si no se especifica un nombre de biblioteca ni SYSLIB. (Esta opción es la letra mayúscula *I*, no la letra minúscula *l*.)

Sólo se permite una única vía de acceso para cada opción `-I` . Para añadir varias vías de acceso, utilice varias opciones `-I` . No inserte espacios entre `-I` y *xxx*.

-m

Genera automáticamente información de dependencia que se puede utilizar en un archivo make.

Con esta opción, `GNU make` puede recompilar automáticamente un programa COBOL si se modifica su fuente o cualquiera de los libros de copias que utiliza. Esto mejora la productividad del desarrollador al permitir a los desarrolladores realizar compilaciones incrementales e iterativas, al mismo tiempo que se asegura de que los programas se compilen siempre con la versión más reciente de cualquier libro de copias que utilicen.

Consulte la documentación de GNU make (gmake) para obtener detalles sobre cómo utilizar esta característica en un archivo make en Generación automática de requisitos previos.

-q*xxx*
Pasa opciones al compilador, donde *xxx* es cualquier opción de compilador o conjunto de opciones de compilador. No inserte espacios entre `-q` y *xxx*.

Si un paréntesis forma parte de la opción o subopción del compilador, o si se especifica una serie de opciones, inclúyalas entre comillas.

Para especificar varias opciones, delimite cada opción con un espacio en blanco o una coma. Por ejemplo, las dos series de opciones siguientes son equivalentes:

```
-qoptiona,optionb
```

```
-q"optiona optionb"
```

Si tiene previsto utilizar un script de shell para automatizar las tareas de `cob2` , se proporciona una sintaxis especial para la opción `-q`*xxx* . Para obtener detalles, consulte la tarea relacionada sobre la compilación utilizando scripts de shell.

Opciones que se aplican al enlace

-cmain
(Tiene un efecto sólo si también especifica `-host`.) Hace que un archivo de objeto C (que contiene una rutina principal) sea el punto de entrada principal en el archivo ejecutable. En C, una rutina principal se identifica por el nombre de función main ().

Si enlaza un archivo de objeto C que contiene una rutina principal con uno o más archivos de objeto COBOL, debe utilizar `-cmain` para designar la rutina C como punto de entrada principal. Un programa COBOL no puede ser el punto de entrada principal en un archivo ejecutable que contiene una rutina principal C . Se produce un comportamiento imprevisible si se intenta y no se emite ningún diagnóstico.

-main:*xxx*
Hace que *xxx* sea el primer archivo de la lista de archivos pasados al enlazador. La finalidad de esta opción es convertir el archivo especificado en el programa principal del archivo ejecutable. *xxx* debe identificar de forma exclusiva el archivo de objeto o la biblioteca de archivado, y el sufijo debe ser .o o .a, respectivamente.

Si no se especifica `-main` , el primer objeto, biblioteca de archivado o archivo fuente especificado en el mandato es el primer archivo de la lista de archivos pasados al enlazador.

Si la sintaxis de `-main:`*xxx* no es válida, o si *xxx* no es el nombre de un objeto o archivo fuente procesado por el mandato, el mandato termina.

-o *xxx*
Nombra el módulo ejecutable *xxx*, donde *xxx* es cualquier nombre. Si no se utiliza la opción `-o` , el nombre del módulo ejecutable toma como valor predeterminado a.out.

Opciones que se aplican a la compilación y al enlace

-F*xxx*
Utiliza *xxx* como un archivo de configuración o una stanza en lugar de los valores predeterminados especificados en el archivo de configuración /opt/ibm/cobol/1.2.0/etc/cob2.cfg . *xxx* tiene uno de los formatos siguientes:

- *archivo_configuración*:*stanza*
- *archivo_configuración*
- :*stanza*

-g
Produce información simbólica utilizada por el depurador. Establece la opción de compilador TEST.

-q32
Especifica que se va a generar un programa objeto de 32 bits. Establece la opción de compilador `ADDR(32)` . Establece el `-m32 linker option`, que indica al enlazador que cree un módulo ejecutable de 32 bits.

-q64
Especifica que se va a generar un programa objeto de 64 bits. Establece la opción de compilador `ADDR(64)` . Establece la opción de enlazador de `-m64` , que indica al enlazador que cree un módulo ejecutable de 64 bits.

-v
Muestra los pasos de compilación y enlace, y los ejecuta.

-#
Muestra los pasos de compilación y enlace, pero no los ejecuta.

-?, ?
Muestra ayuda para el mandato `cob2` .

Tareas relacionadas
“Compilación desde la línea de mandatos” en la página 240
“Compilación utilizando archivos de proceso por lotes o archivos de mandatos de scripts de shell” en la página 241
“Pasar opciones al enlazador” en la página 251
“Utilización de argumentos de línea de mandatos” en la página 486

Referencias relacionadas
“Variables de entorno de compilador” en la página 233
“ADDR” en la página 268

Enlace de programas

Utilice el enlazador para enlazar archivos de objeto especificados y crear un archivo ejecutable o un objeto compartido.

Acerca de esta tarea

Usted tiene una opción de maneras de iniciar el enlazador. Puede utilizar:

- El mandato `cob2` :

  Este mandato llama al enlazador a menos que especifique la opción `-c` .

  `cob2` enlaza con las bibliotecas multihebra COBOL, por lo que no es necesario ninguna opción de hebra adicional.

  Enlace con C: El enlazador acepta archivos .o, pero no acepta archivos .c. Si desea enlazar archivos C y COBOL, primero genere archivos .o para los archivos de origen C utilizando el mandato `gcc` .
- Un archivo make:

  Puede utilizar un archivo make para organizar la secuencia de acciones (como compilar y enlazar) que son necesarias para crear el programa. En el archivo make, puede utilizar sentencias de enlazador para especificar el tipo de salida que necesita.

Puede especificar opciones de enlazador utilizando cualquiera de los métodos descritos anteriormente.

“Ejemplos: utilización de cob2 para enlazar” en la página 251

Tareas relacionadas
“Compilación desde la línea de mandatos” en la página 240
“Pasar opciones al enlazador” en la página 251
Capítulo 24, “Utilización de bibliotecas compartidas”, en la página 489

Referencias relacionadas
“Opciones cob2” en la página 248
“Archivos de entrada y salida de enlazador” en la página 252
“Reglas de búsqueda de enlazador” en la página 252

Pasar opciones al enlazador

Puede especificar opciones de enlazador de cualquiera de estas maneras: un mandato de invocación (cob2) o las sentencias makefile.

Acerca de esta tarea

Las opciones que especifique en un mandato de invocación que no sean reconocidas por el mandato se pasan al enlazador.

Varias opciones de mandato de invocación influyen en el enlace del programa. Para obtener detalles sobre el conjunto de opciones para un mandato determinado, consulte la documentación de dicho mandato.

La tabla siguiente muestra las opciones que es más probable que necesite para los programas COBOL. Preceda cada opción con un guión (-) tal como se muestra.

Tabla 27. ***Opciones de enlazador común***	
Opción	**objetivo**
-L*dir*	Especifica el directorio en el que buscar las bibliotecas especificadas por la opción -l (valor predeterminado: /usr/lib)
-l*name*	Busca el archivo de biblioteca especificado, donde *nombre* selecciona el archivo lib*nombre*.a

"Ejemplos: utilización de cob2 para enlazar" en la página 251

Referencias relacionadas
"Opciones cob2" en la página 248
"Archivos de entrada y salida de enlazador" en la página 252
"Reglas de búsqueda de enlazador" en la página 252

Ejemplos: utilización de cob2 para enlazar

Puede utilizar cualquiera de los mandatos de invocación COBOL para compilar y enlazar los programas. Los ejemplos siguientes ilustran el uso de cob2.

- Para enlazar dos archivos después de compilarlos, no utilice la opción -c . Por ejemplo, para compilar y enlazar alpha.cbl y beta.cbl y generar a.out, especifique:

```
cob2 alpha.cbl beta.cbl
```

  Este mandato crea alpha.o y beta.oy, a continuación, enlaza alpha.o, beta.oy las bibliotecas COBOL. Si el paso de enlace es satisfactorio, genera un programa ejecutable denominado a.out y suprime alpha.o y beta.o.
- Para enlazar un archivo compilado con una biblioteca, entre:

```
cob2 zog.cbl -lmylib
```

  Este mandato hace que el enlazador busque la biblioteca libmylib.so en primer lugar y, a continuación, el archivo de biblioteca de archivado libmylib.a en cada directorio de la vía de acceso de búsqueda de forma consecutiva hasta que se encuentre alguno de ellos.
- Para utilizar opciones para limitar la búsqueda de una biblioteca, especifique:

```
cob2 zog.cbl  -llib1  -llib2
```

En este caso, para satisfacer la primera especificación de biblioteca, el enlazador busca primero la biblioteca `liblib1.so` y, a continuación, el archivo de biblioteca de archivado `liblib1.a` en cada directorio (tal como se describe en el ejemplo anterior). Sin embargo, al mismo tiempo, el enlazador sólo busca `liblib2.a` en esas mismas bibliotecas.

- Para compilar y enlazar en pasos separados, especifique mandatos como los siguientes:

```
cob2 -c file1.cbl               # Produce one object file
cob2 -c file2.cbl file3.cbl     # Or multiple object files
cob2 file1.o file2.o file3.o    # Link with appropriate libraries
```

"Ejemplos: utilización de cob2 para compilar" en la página 241

Archivos de entrada y salida de enlazador

El enlazador toma archivos de objeto, los enlaza entre sí y con cualquier archivo de biblioteca que especifique, y produce un archivo de salida ejecutable. La salida ejecutable puede ser un archivo de programa ejecutable o un objeto compartido.

Entradas de enlazador:

- Opciones
- Archivos de objeto (*.o)
- Archivos de biblioteca de archivado (*.a)
- Archivos de biblioteca dinámica (*.so)

Salidas de enlazador:

- Archivo ejecutable (a.out de forma predeterminada)
- Objeto compartido
- Código de retorno

Archivos de biblioteca: *Las bibliotecas* son archivos que tienen el sufijo .a o .so. Para designar una biblioteca, puede especificar un nombre de vía de acceso absoluta o relativa o utilizar la opción `-l` (letra L en minúsculas) con el formato `-l`*name*. El último formulario designa el archivo lib*nombre*.a, o en modo dinámico, archivo lib*nombre*.so, que se debe buscar en varios directorios. Estos directorios de búsqueda incluyen los directorios que especifique utilizando las opciones `-L` y los directorios de biblioteca estándar /usr/lib y /lib.

La variable de entorno LD_LIBRARY_PATH no se utiliza para buscar bibliotecas que especifique en la línea de mandatos de forma explícita (por ejemplo, `libc.a`) o utilizando la opción `-l` (por ejemplo, `-lc`). Debe utilizar las opciones `-L`*dir* para indicar los directorios en los que se deben buscar las bibliotecas que ha especificado con una opción `-l` .

Puede crear archivos de biblioteca combinando uno o más archivos en un único archivo de archivado utilizando Linux `ar` .

"Ejemplo: creación de una biblioteca compartida de ejemplo" en la página 490

Tareas relacionadas
"Pasar opciones al enlazador" en la página 251

Referencias relacionadas
"Reglas de búsqueda de enlazador" en la página 252
"Valores predeterminados de nombre de archivo de enlazador" en la página 253

Reglas de búsqueda de enlazador

Al buscar un archivo de objeto (.o) o un archivo de biblioteca de archivado (.a), el enlazador busca en varias ubicaciones hasta que se satisface la búsqueda.

El enlazador busca estas ubicaciones:

1. El directorio que especifique para el archivo

   Si especifica una vía de acceso con el archivo, el enlazador sólo busca esa vía de acceso y deja de enlazar si el archivo no se puede encontrar allí.
2. El directorio actual, si no ha especificado una vía de acceso
3. El valor de la variable de entorno LD_LIBRARY_PATH, si se ha definido

Si utiliza bibliotecas que no sean las predeterminadas en /usr/lib, puede especificar una o más opciones `-L` que apunten a las ubicaciones de las otras bibliotecas. También puede establecer la variable de entorno LD_LIBRARY_PATH , que le permite especificar una vía de acceso de búsqueda para bibliotecas en tiempo de ejecución.

Si el enlazador no puede localizar un archivo, genera un mensaje de error y detiene el enlace.

Tareas relacionadas
"Pasar opciones al enlazador" en la página 251

Referencias relacionadas
"Valores predeterminados de nombre de archivo de enlazador" en la página 253

Valores predeterminados de nombre de archivo de enlazador

Si no especifica un nombre de archivo, el enlazador asume los nombres predeterminados.

Tabla 28. ***Nombres de archivo predeterminados asumidos por el enlazador***

Archivo	Nombre de archivo predeterminado	Sufijo predeterminado
Archivos de objeto	Ninguna. Debe especificar al menos un nombre de archivo de objeto.	.o
Archivo de salida	a.out	Ninguna
Archivos de biblioteca	Las bibliotecas predeterminadas definidas en los archivos de objeto. Utilice las opciones del compilador para definir las bibliotecas por omisión. Las bibliotecas adicionales que especifique se buscan antes que las bibliotecas predeterminadas.	.a o .so

Corrección de errores en el enlace

Cuando se utiliza la opción de compilador `PGMNAME(UPPER)` , los nombres de los subprogramas a los que se hace referencia en las sentencias `CALL` se convierten a mayúsculas. Este cambio afecta al enlazador, que reconoce nombres sensibles a mayúsculas y minúsculas.

Acerca de esta tarea

Por ejemplo, el compilador convierte `Call "RexxStart"` en `Call "REXXSTART"`. Si el nombre real del programa llamado es RexxStart, el enlazador no encontrará el programa y generará un mensaje de error que indica que REXXSTART es una referencia externa no resuelta.

Este tipo de error normalmente se produce cuando se llama a rutinas de API proporcionadas por otro producto de software. Si las rutinas de API tienen nombres con mayúsculas y minúsculas, debe realizar las dos acciones siguientes:

- Utilice la opción de compilador `PGMNAME(MIXED)` .

- Asegúrese de que las sentencias CALL especifiquen la combinación correcta de mayúsculas y minúsculas en los nombres de las rutinas de API.

ejecutar programas

Para ejecutar un programa COBOL, establezca variables de entorno, emita el mandato para ejecutar el ejecutable y, a continuación, corrija los errores de tiempo de ejecución.

Acerca de esta tarea

Procedimiento

1. Asegúrese de que se hayan establecido las variables de entorno necesarias.

   Por ejemplo:

   - Si el programa utiliza un nombre de variable de entorno para asignar un valor a un nombre de archivo del sistema, establezca esa variable de entorno.
   - Si el programa utiliza opciones de tiempo de ejecución, especifique sus valores en la variable de entorno de ejecución COBRTOPT.
2. Ejecute el programa: En la línea de mandatos, especifique el nombre del módulo ejecutable o ejecute un archivo de mandatos que invoque dicho módulo. Incluir los argumentos de programa necesarios en la línea de mandatos.

   Por ejemplo, si el mandato `cob2 alpha.cbl beta.cbl-o gamma` es satisfactorio, puede ejecutar el programa especificando `gamma`.
3. Corrija los errores de tiempo de ejecución.

   Puede utilizar el depurador para examinar el estado del programa en el momento de los errores.

Resultados

Consejo: Si los mensajes de tiempo de ejecución están abreviados o incompletos, es posible que las variables de entorno LANG o NLSPATH o ambas se hayan establecido incorrectamente.

Tareas relacionadas
“Establecimiento de variables de entorno” en la página 229
“Depuración utilizando IBM Debug for Linux en x86” en la página 328
“Utilización de argumentos de línea de mandatos” en la página 486

Referencias relacionadas
Capítulo 15, “Opciones de tiempo de ejecución”, en la página 315

Apéndice F, “Mensajes de tiempo de ejecución”, en la página 625

Capítulo 13. Especificación de opciones de compilador en la línea de mandatos

La mayoría de las opciones especificadas en la línea de mandatos alteran temporalmente los valores predeterminados de la opción y las opciones establecidas en el archivo de configuración.

Existen dos tipos de opciones de línea de mandatos:

- Opciones de distintivo
- **-q***palabra_opción* (específico del compilador)

Conceptos relacionados
“Opciones de distintivo” en la página 255
“Opciones -q” en la página 264

Tareas relacionadas
“Compilación de programas” en la página 239

Opciones de distintivo

COBOL for Linux da soporte a una serie de opciones de distintivo convencionales comunes que se utilizan en sistemas Linux . Las opciones de distintivo distinguen entre mayúsculas y minúsculas y se aplican a el mandato de invocación de `cob2`.

COBOL for Linux también da soporte a distintivos que se dirigen a otras herramientas de programación y programas de utilidad (por ejemplo, el mandato `scu`).

Algunas opciones de distintivo tienen argumentos que forman parte del distintivo. Por ejemplo:

```
cob2 stem.cbl -F/home/tools/test3/new.cfg:cob2
```

donde `new.cfg` es un archivo de configuración personalizado.

Puede especificar distintivos que no toman argumentos en una serie. Por ejemplo:

```
cob2 -ocv file.cbl
```

```
cob2 -o -c -v file.cbl
```

Una opción de distintivo que toma argumentos se puede especificar como parte de una sola serie, pero sólo puede utilizar un distintivo que tome argumentos, y debe ser la última opción especificada. Por ejemplo, puede utilizar el distintivo `-o` (para especificar un nombre para el archivo ejecutable) junto con otros distintivos, sólo si la opción `-o` y su argumento se especifican en último lugar. Por ejemplo:

```
cob2 -ocv test test.cbl
```

tiene el mismo efecto que:

```
cob2 -o -c -vtest test.cbl
```

La mayoría de las opciones de distintivo son una sola letra, pero algunas son dos letras. Tenga cuidado de no especificar dos o más opciones en una sola serie si hay otra opción que utiliza esa combinación de letras.

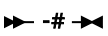
Conceptos relacionados
Capítulo 13, “Especificación de opciones de compilador en la línea de mandatos”, en la página 255
“Opciones -q” en la página 264

-# (signo de libra)

objetivo

Previsualiza los pasos de compilación que se especifican en la línea de mandatos, sin invocar realmente ningún componente del compilador. Cuando esta opción está habilitada, la información se graba en la salida estándar, mostrando los nombres de los programas dentro del preprocesador, compilador y enlazador que se invocarían, y las opciones por omisión que se especificarían para cada programa. El preprocesador, compilador y enlazador no se invocan. Esta opción se aplica a la compilación y al enlace.

Sintaxis



Valores predeterminados

El compilador no muestra el progreso de la compilación.

Uso

Puede utilizar este mandato para determinar los mandatos y archivos implicados en una compilación determinada. Evita la sobrecarga de compilar el código fuente y sobrescribir los archivos existentes, como los archivos `.lst` . Esta opción muestra la misma información que `-v`, pero no invoca el compilador. La opción `-#` altera temporalmente la opción `-v` .

Referencias relacionadas

"-v" en la página 263

-?, ?

objetivo

Muestra ayuda para el mandato `cob2` . Esta opción se aplica a la compilación y al enlace.

Sintaxis

Valores predeterminados

El compilador no muestra la información de ayuda del mandato.

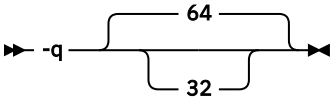
-q32, -q64

objetivo

`-q32`: especifica que se va a generar un programa objeto de 32 bits. Establece la opción de compilador `ADDR(32)` . Establece la opción de enlazador de `-m32` , que indica al enlazador que cree un módulo ejecutable de 32 bits.

`-q64`: Especifica que se va a generar un programa de objeto de 64 bits. Establece la opción de compilador ADDR (64). Establece la opción de enlazador `-m64` , que indica al enlazador que cree un módulo ejecutable de 64 bits.

Sintaxis



Valores predeterminados

-q64

Referencias relacionadas

"ADDR" en la página 268

-c

objetivo

Impide que el objeto completado se envíe al enlazador. Cuando esta opción está en vigor, el compilador crea un archivo de objeto de salida, *file_name*.o. Esta opción sólo se aplica a la compilación.

Sintaxis

Valores predeterminados

De forma predeterminada, el compilador invoca el enlazador para enlazar archivos de objeto en un archivo ejecutable final.

Ejemplos

- Para compilar un archivo denominado alpha.cbl, especifique:

```
cob2 -c alpha.cbl
```

  El archivo compilado se denomina alpha.o.
- Para compilar dos archivos que se denominan alpha.cbl y beta.cbl, especifique:

```
cob2 -c alpha.cbl beta.cbl
```

  Los archivos compilados se denominan alpha.o y beta.o.
- Para enlazar dos archivos, compílelos sin la opción -c . Por ejemplo, para compilar y enlazar alpha.cbl y beta.cbl y generar gamma, especifique:

```
cob2 alpha.cbl beta.cbl -o gamma
```

  Este mandato crea alpha.o y beta.oy, a continuación, enlaza alpha.o, beta.oy las bibliotecas COBOL. Si el paso de enlace es satisfactorio, genera un programa ejecutable denominado gamma.
- Para compilar alpha.cbl con las opciones LIST y NODATA , especifique:

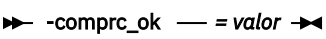
```
cob2 -qlist,noadata alpha.cbl
```

-comprc_ok

objetivo

Controla el comportamiento tras la devolución del compilador. Si el código de retorno es menor o igual que *n*, el mandato continúa en el paso de enlace, o en el caso de sólo compilación, sale con un código de retorno cero. Si el código de retorno generado por el compilador es mayor que *n*, el mandato sale con el mismo código de retorno devuelto por el compilador. Esta opción sólo se aplica a la compilación.

Sintaxis



Valores predeterminados

El valor predeterminado es `-comprc_ok=4`.

Uso

Cuando esta opción está en vigor, el compilador crea un archivo de objeto de salida, *`file_name`*`.o`.

-dll | -dso | -shared

objetivo

Cambia la salida del editor de enlaces del PIE (Position Independent Executable) predeterminado a un objeto compartido dinámicamente (DSO).

Sintaxis

Valores predeterminados

La salida del editor de enlaces es un PIE.

Uso

Un PIE se puede invocar como cualquier otro programa ejecutable, pero no se puede utilizar como destino de una llamada dinámica COBOL o una transacción CICS .

Un DSO es lo contrario. No se puede llamar desde la línea de mandatos, pero se puede utilizar como destino de una llamada dinámica COBOL o una transacción CICS .

-F

objetivo

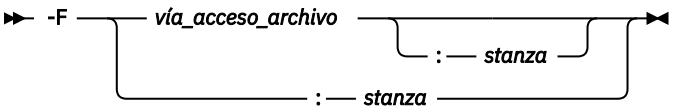
Utiliza *xxx* como un archivo de configuración o una stanza en lugar de los valores predeterminados especificados en el archivo de configuración de `/opt/ibm/cobol/1.2.0/etc/cob2.cfg` . *xxx* tiene uno de los formatos siguientes:

- *archivo_configuración*:*stanza*

- *archivo_configuración*
- :*stanza*

Esta opción se aplica a la compilación y al enlace.

Sintaxis



Valores predeterminados

De forma predeterminada, el compilador utiliza el archivo de configuración que se proporciona en el momento de la instalación y utiliza la stanza definida en dicho archivo para el mandato de invocación que se está utilizando actualmente.

parámetros

File_path

El nombre completo de la vía de acceso del archivo de configuración del compilador alternativo que se va a utilizar.

stanza

El nombre de la stanza del archivo de configuración que se va a utilizar para la compilación. Esto indica al compilador que utilice las entradas bajo ese `stanza` independientemente del mandato de invocación que se esté utilizando.

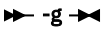
Referencias relacionadas
"Stanzas en el archivo de configuración" en la página 244

-g

Finalidad

Genera información utilizada por el depurador. Establece la opción de compilador TEST. Esta opción se aplica a la compilación y al enlace.

Sintaxis



Valores predeterminados

No genera información de depuración. No se conserva ningún estado de programa.

Ejemplos

Utilice el mandato siguiente para compilar `myprogram.cbl` y generar un programa ejecutable llamado testing para la depuración:

```
cob2 myprogram.cbl -o testing -g
```

Tareas relacionadas
"Depuración utilizando IBM Debug for Linux en x86" en la página 328

Referencias relacionadas
“TEST” en la página 301

-host

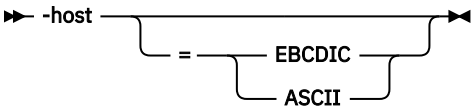
objetivo

Establece estas opciones de compilador para la representación de datos COBOL de host y la semántica de lenguaje:

- `BINARY(BE)`
- `CHAR(EBCDIC)`
- `COLLSEQ(EBCDIC)`
- `FLOAT(BE)`
- `NCOLLSEQ(BIN)`
- `UTF16(BE)`

La opción `-host` cambia el formato de los argumentos de línea de mandatos del programa COBOL de una matriz de punteros a una serie de caracteres EBCDIC que tiene un prefijo de media palabra que contiene la longitud de la serie. Para obtener información adicional, consulte la tarea relacionada siguiente sobre el uso de argumentos de línea de mandatos.

Sintaxis



Valores predeterminados

El compilador no establece opciones de compilador para la representación de datos COBOL de host y la semántica de lenguaje.

parámetros

EBCDIC
Igual que `-host`. Pasa una lista de parámetros de estilo z/OS en EBCDIC al programa.

ASCII
Pasa una lista de parámetros de estilo z/OS en ASCII al programa.

Referencias relacionadas
“BINARIO” en la página 271
CHAR
COLLSEQ
FLOTANTE
NCOLLSEQ
“UTF16” en la página 304

-I

objetivo

Añade la vía de acceso *xxx* a los directorios en los que se buscarán los libros de copias si no se especifica un nombre de biblioteca ni SYSLIB. (Esta opción es la letra mayúscula *I*, no la letra minúscula *l*.)

Sólo se permite una única vía de acceso para cada opción `-I` . Para añadir varias vías de acceso, utilice varias opciones `-I` .

Sintaxis

parámetros

vía_acceso_directorio

La vía de acceso del directorio donde el compilador debe buscar los archivos de cabecera.

Valores predeterminados

No hay ningún directorio predeterminado para buscar libros de copias.

Uso

Si se especifica la opción de directorio `-I` tanto en el archivo de configuración como en la línea de mandatos, primero se busca en las vías de acceso especificadas en el archivo de configuración. La opción de directorio `-I` se puede especificar más de una vez en la línea de mandatos. Si especifica más de una opción `-I` , se busca en los directorios en el orden en que aparecen en la línea de mandatos. La opción `-I` no tiene ningún efecto en los archivos que se incluyen utilizando un nombre de vía de acceso absoluta

Ejemplos

Para compilar `myprogram.cbl` y buscar `/usr/tmp and then /oldstuff/history` los archivos incluidos, especifique:

```
cob2 myprogram.cbl -I/usr/tmp -I/oldstuff/history
```

-M

Finalidad

Utilice la opción de compilador **-M** para generar automáticamente información de dependencia que se puede utilizar en un archivo make.

Sintaxis

Valores predeterminados

De forma predeterminada, el compilador no genera información de dependencia.

Uso

Con esta opción, GNU make puede recompilar automáticamente un programa COBOL si se modifica su fuente o cualquiera de los libros de copias que utiliza. Esto mejora la productividad del desarrollador al permitir a los desarrolladores realizar compilaciones incrementales e iterativas, al mismo tiempo que se asegura de que los programas se compilen siempre con la versión más reciente de cualquier libro de copias que utilicen.

Consulte la documentación de GNU make (gmake) para obtener detalles sobre cómo utilizar esta característica en un archivo make en Generación automática de requisitos previos.

Ejemplo

Si tiene un archivo `main.cbl` que incluye dos programas donde cada uno incluye un libro de copias, y lo compila con el mandato siguiente:

```
cob2 -c -I./copybooks -M main.cbl
```

, generará el siguiente mensaje:

```
Generating dependencies...
Compiling...
IBM COBOL para Linux 1.2.0 compile started
End of compilation 1, program PGM1, no statements flagged.
End of compilation 2, program PGM2, no statements flagged.
```

Si el mandato es satisfactorio, se creará un archivo `main.d` que contendrá una lista de dependencias y el código de retorno del compilador será 0. A continuación se muestra un ejemplo del archivo `main.d` :

```
main.o: main.cbl \
/home/username/copybook1.cbl \
/home/username/./copybooks/copybook2.cbl \
/home/username/copybook1.cbl \
/home/username/./copybooks/copybook2.cbl
```

En el ejemplo anterior había dos programas en `main.cbl` y ambos incluían el mismo libro de copias. Por lo tanto, `main.d` muestra dos entradas para los mismos libros de copias. Cualquier duplicado será ignorado por la utilidad make de GNU.

Si durante la generación de dependencias el código de retorno del compilador es mayor que el valor de `-comprc_ok`, se generarán los mensajes del compilador y la compilación saldrá porque se considera que la generación de dependencias ha fallado.

-principal

objetivo

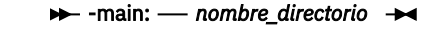
Hace que *xxx* sea el primer archivo de la lista de archivos que se pasan al enlazador. La finalidad de esta opción es convertir el archivo especificado en el programa principal del archivo ejecutable. *xxx* debe identificar de forma exclusiva el archivo de objeto o la biblioteca de archivado, y el sufijo debe ser `.o` o `.a`, respectivamente.

Si no se especifica `-main` , el primer objeto, biblioteca de archivado o archivo fuente especificado en el mandato es el primer archivo de la lista de archivos que se pasan al enlazador.

Si la sintaxis de `-main:`*xxx* no es válida, o si *xxx* no es el nombre de un objeto o archivo de origen procesado por el mandato, el mandato termina.

Esta opción sólo se aplica al enlace.

Sintaxis



Valores predeterminados

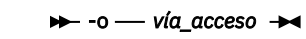
No se ha especificado la opción `-main` .

-o

objetivo

Nombra el programa ejecutable o la biblioteca compartida *xxx*, donde *xxx* es cualquier nombre. Si no se utiliza la opción `-o` , el nombre del módulo ejecutable toma como valor predeterminado a `.out`. Esta opción sólo se aplica al enlace.

Sintaxis



parámetros

path

Cuando se utiliza la opción para compilar desde archivos de origen, *vía_acceso* puede ser el nombre de un archivo o directorio. La *vía de acceso* puede ser un nombre de vía de acceso relativa o absoluta. Cuando se utiliza la opción para enlazar desde archivos de objeto, *vía de acceso* debe ser un nombre de archivo. Si *vía_acceso* es el nombre de un directorio existente, los archivos creados por el compilador se colocan en ese directorio. Si *vía_acceso* no es un directorio existente, *vía_acceso* es el nombre del archivo generado por el compilador. Consulte a continuación los ejemplos

Valores predeterminados

Si especifica la opción `-c` , se genera un archivo de objeto de salida, *`file_name`*`.o`, para cada archivo de entrada. El enlazador no se invoca y los archivos de objeto se colocan en el directorio actual. Todo el proceso se detiene al finalizar la compilación. El compilador proporciona a los archivos de objeto un sufijo `.o` , por ejemplo, *`file_name`*`.o`, a menos que especifique la opción `-o` , lo que proporciona un sufijo diferente o ningún sufijo.

Uso

Si utiliza la opción `-c` con `-o` juntos y la vía de acceso no es un directorio existente, sólo puede compilar un archivo de origen a la vez. En este caso, si se lista más de un nombre de archivo de origen en la invocación del compilador, el compilador emite un mensaje de aviso e ignora `-o`.

Ejemplos

Para compilar `myprogram.cbl` para que el ejecutable resultante se denomine `myaccount`, suponiendo que no exista ningún directorio con el nombre `myaccount` , especifique:

```
cob2 myprogram.cbl -o myaccount
```

Para compilar `test.cbl` sólo en un archivo de objeto y asignar el nombre `new.o`al archivo de objeto, entre:

```
cob2 test.cbl -c -o new.o
```

Referencias relacionadas

-v

objetivo

Muestra los pasos de compilación y enlace, y los ejecuta. Esta opción se aplica a la compilación y al enlace. Cuando la opción `-v` está en vigor, la información se muestra en una lista separada por comas. Esta opción se aplica a la compilación y al enlace.

Nota: La opción -v se altera temporalmente mediante la opción *-#* .

Sintaxis

Valores predeterminados

El compilador no muestra el progreso de la compilación.

Ejemplos

Para compilar `myprogram.cbl` para poder ver el progreso de la compilación y ver los mensajes que describen el progreso de la compilación, los programas que se invocan y las opciones que se especifican, entre:

```
cob2 myprogram.cbl -v
```

Referencias relacionadas

"-# (signo de libra)" en la página 256

Opciones -q

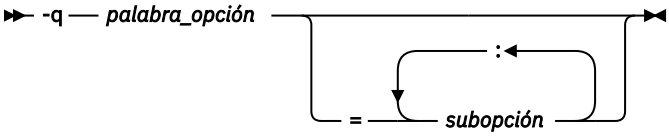
Pasa opciones al compilador, donde *xxx* es cualquier opción de compilador o conjunto de opciones de compilador. No inserte espacios entre -q y *xxx*. Si un paréntesis forma parte de la opción o subopción del compilador, o si se especifica una serie de opciones, inclúyalas entre comillas. Si un paréntesis forma parte de la opción o subopción del compilador, o si se especifica una serie de opciones, inclúyalas entre comillas.

Para especificar varias opciones, delimite cada opción con un espacio en blanco o una coma. Por ejemplo, las dos series de opciones siguientes son equivalentes:

```
-qoptiona,optionb
```

```
-q"optiona optionb"
```

Si tiene previsto utilizar un script de shell para automatizar las tareas de `cob2` , se proporciona una sintaxis especial para la opción *-qxxx* . Para obtener detalles, consulte la tarea relacionada sobre la compilación utilizando scripts de shell.



Conceptos relacionados

Capítulo 13, "Especificación de opciones de compilador en la línea de mandatos", en la página 255

Referencias relacionadas

"opciones de compilador" en la página 265
"Opciones de compilador en conflicto" en la página 267

Tareas relacionadas

Compilación utilizando scripts de shell

opciones de compilador

Puede dirigir y controlar la compilación utilizando opciones de compilador o utilizando sentencias de dirección de compilador (directivas de compilador).

Las opciones de compilador afectan a los aspectos del programa que se listan en la tabla siguiente. La información enlazada para cada opción proporciona la sintaxis para especificar la opción y describe la opción, sus parámetros y su interacción con otros parámetros.

Tabla 29. ***opciones de compilador***

Aspecto del programa	Opción de compilador	Valor predeterminado	Abreviaturas de opción
Idioma de origen	"HARIT" en la página 270	ARITH(COMPAT)	AR(C|E)
	"CICS" en la página 274	NOCICS	Ninguna
	"MONEDA" en la página 277	NOCURRENCY	CURR|NOCURR
	"SÍMBOLO" en la página 290	NSYMBOL(NATIONAL)	NS(NAT|DBCS)
	"NÚMERO" en la página 291	NONUMBER	NUM|NONUM
	"APOST/CITA" en la página 293	QUOTE	Q|APOST
	"SECUENCIA" en la página 295	SEQUENCE	SEQ|NOSEQ
	"SOSI" en la página 295	NOSOSI	Ninguna
	"SQL" en la página 298	SQL("")	Ninguna
	"FORMATOORIGEN" en la página 298	SRCFORMAT(COMPAT)	SF(C|E)
Proceso de fecha	"DATEPROC" en la página 277	NODATEPROC, o DATEPROC(FLAG) si sólo se especifica DATEPROC	DP|NODP
	"FECHA Y HORA" en la página 278	DATETIME (1900 40)	Ninguna
	"VENTANA" en la página 307	YEARWINDOW(1900)	YW
Mapas y listados	"LINECOUNT" en la página 287	LINECOUNT(60)	LC
	"lista" en la página 287	NOLIST	Ninguna
	"LSTFILE" en la página 288	LSTFILE(LOCALE)	LST
	"MAP" en la página 288	NOMAP	Ninguna
	"fuente" en la página 297	SOURCE	S|NOS
	"ESPACIO" en la página 297	SPACE(1)	Ninguna
	"TERMINAL" en la página 301	TERMINAL	TERM|NOTERM
	"VBREF" en la página 305	NOVBREF	Ninguna
	"XREF" en la página 306	XREF(FULL)	X|NOX
Generación de módulo de objeto	"COMPILAR" en la página 276	NOCOMPILE(S)	C|NOC
	"PGMNAME" en la página 292	PGMNAME(UPPER)	PGMN(LU|LM)
	"SEPOBJ" en la página 294	SEPOBJ	Ninguna

*Tabla 29. **opciones de compilador** (continuación)*

Aspecto del programa	Opción de compilador	Valor predeterminado	Abreviaturas de opción
Control de código de objeto	"ADDR" en la página 268	ADDR(64)	Ninguna
	"BINARIO" en la página 271	BINARY(NATIVE)	Ninguna
	"char" en la página 272	CHAR(NATIVE)	Ninguna
	"COLLSEQ" en la página 275	COLLSEQ(BIN)	CS(L|E|BIN|B)
	"DEFECTO" en la página 279	NODEFINE	DEF | NODEF
	"DIAGTRUNC" en la página 280	NODIAGTRUNC	DTR|NODTR
	"FLOAT" en la página 286	FLOAT(NATIVE)	Ninguna
	"NCOLLSEQ" en la página 290	NCOLLSEQ(BINARY)	NCS(L|BIN|B)
	"OPTIMIZAR" en la página 291	NOOPTIMIZE	OPT|NOOPT
	"TRUNC" en la página 302	TRUNC(STD)	Ninguna
	"ZWB" en la página 307	ZWB	Ninguna
Comportamiento de la sentencia CALL	"DYNAM" en la página 281	NODYNAM	DYN|NODYN
Depuración y diagnóstico	"BANDERA" en la página 284	FLAG(I,I)	F|NOF
	"FLAGSTD" en la página 285	NOFLAGSTD	Ninguna
	"SRANGE" en la página 299	NOSSRANGE	SSR(MSG|ABD)|NOSSR
	"TEST" en la página 301	NOTEST	Ninguna
Otros	"DATOS" en la página 268	NOADATA	Ninguna
	"CALLINT" en la página 271	CALLINT(SYSTEM,NODESC)	Ninguna
	"salida" en la página 282	NOEXIT	NOEX|EX(INX|NOINX, LIBX|NOLIBX, PRTX|NOPRTX, ADX|NOADX, MSGX|NOMSGX)
	"MDECK" en la página 289	NOMDECK	NOMD|MD|MD(C|NOC)
	"SPILL" en la página 297	SPILL(512)	Ninguna
	"HEBRA" en la página 302	NOTHREAD	Ninguna
	"WCLEAR" en la página 305	NOWSCLEAR	Ninguna

Valores predeterminados de instalación: Los valores predeterminados listados para las opciones anteriores son los valores predeterminados que se suministran con el producto.

Especificaciones de opciones:

- Las opciones y subopciones del compilador no distinguen entre mayúsculas y minúsculas.
- Para las opciones de compilador que van seguidas de argumentos como subopciones, debe seguir utilizando el formato de `option(argument)`, en lugar de especificar `option=argument`.

Consideraciones sobre el rendimiento: Las opciones de compilador ADDR, ARITH, CHAR, DYNAM, FLOAT, OPTIMIZE, SSRANGE, TEST, TRUNCy WSCLEAR pueden afectar al rendimiento del tiempo de ejecución.

Tareas relacionadas
“Compilación de programas” en la página 239
Capítulo 27, “Ajuste del programa”, en la página 525

Referencias relacionadas
Capítulo 14, “Sentencias de direccionamiento de compilador”, en la página 309
“Opciones de compilador relacionadas con el rendimiento” en la página 534

Valores de opción para conformidad con 85 COBOL Standard

Las opciones de compilador y las opciones de tiempo de ejecución son necesarias para la conformidad con el estándar COBOL 85.

Son necesarias las siguientes opciones de compilador:

- DYNAM
- NOCICS
- NOSOSI
- NOHEBRA
- PGMNAME (COMPAT) o PGMNAME (LONGUPPER)
- PRESUPUESTO
- TRUNC (STD)
- ZWB

Puede utilizar la opción de compilador FLAGSTD para señalar elementos no conformes como, por ejemplo, las extensiones IBM .

Opciones de compilador en conflicto

El compilador COBOL para Linux puede encontrar opciones de compilador en conflicto de dos maneras: tanto el formato positivo como el negativo de una opción se especifican en el mismo nivel en la jerarquía de prioridad de las opciones, o se especifican opciones mutuamente excluyentes en el mismo nivel.

El compilador reconoce las opciones en el siguiente orden de prioridad de mayor a menor:

1. Opciones especificadas en la sentencia PROCESS (o CBL)
2. Opciones especificadas en la invocación del mandato `cob2`
3. Opciones establecidas en la variable de entorno COBOPT
4. Opciones establecidas en el atributo compopts del archivo de configuración (`.cfg`)
5. IBM opciones predeterminadas

Si especifica opciones en conflicto en el mismo nivel de la jerarquía, la opción especificada entra en vigor en último lugar.

Si especifica opciones de compilador mutuamente excluyentes en el mismo nivel, el compilador fuerza una de las opciones a un valor no conflictivo y genera un mensaje de error. Por ejemplo, Si especifica CICS y DYNAM en la sentencia PROCESS en cualquier orden, CICS entra en vigor y DYNAM se ignora, tal como se muestra en la tabla siguiente.

Tabla 30. ***Opciones de compilador mutuamente excluyentes***

Especificadas	**Omitidos**	**Forzado el**
`CICS`	`ADDR(64)`	`ADDR(32)`
	`DYNAM`	`NODYNAM`
	`THREAD`	`NOTHREAD`

Sin embargo, las opciones especificadas en un nivel superior de prioridad alteran temporalmente las opciones especificadas en un nivel inferior de prioridad. Por ejemplo, si codifica CICS en la variable de entorno COBOPT pero DYNAM en la sentencia PROCESS, DYNAM entra en vigor porque las opciones codificadas en la sentencia PROCESS y las opciones forzadas por una opción codificada en la sentencia PROCESS tienen una prioridad más alta.

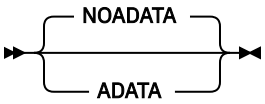
Tareas relacionadas
“Compilación de programas” en la página 239

Referencias relacionadas
“Variables de entorno de compilador” en la página 233
“Stanzas en el archivo de configuración” en la página 244
“Opciones cob2” en la página 248
Capítulo 14, “Sentencias de direccionamiento de compilador”, en la página 309

DATOS

Utilice ADATA cuando desee que el compilador cree un archivo SYSADATA que contenga registros de información de compilación adicional.

Sintaxis de la opción ADATA



El valor predeterminado es: NOADATA

Las abreviaturas son: Ninguna

La información SYSADATA la utilizan otras herramientas, que establecerán ADATA ON para su uso.

El tamaño del archivo SYSADATA generalmente crece con el tamaño del programa asociado.

Especificación de opción: No puede especificar la opción ADATA en una sentencia PROCESS (o CBL). Sólo puede especificarlo de una de las maneras siguientes:

- Como opción en el mandato cob2 o una de sus variantes
- En la variable de entorno COBOPT
- En el atributo compopts del archivo de configuración (.cfg)

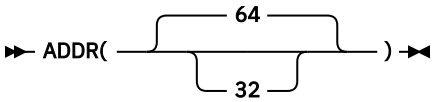
Nota: La opción ADATA no está soportada actualmente. Utilice el valor predeterminado, NOADATA por ahora.

Referencias relacionadas
“Variables de entorno de compilador” en la página 233
“Stanzas en el archivo de configuración” en la página 244
“Opciones cob2” en la página 248

ADDR

Utilice la opción de compilador ADDR para indicar si se debe generar un programa de objeto de 32 bits o de 64 bits.

Sintaxis de la opción ADDR



El valor predeterminado es: ADDR(64)

Las abreviaturas son: Ninguna

Especificación de opción:

Puede especificar la opción ADDR de cualquiera de las maneras en que especifique otras opciones de compilador, tal como se describe en la tarea relacionada sobre la compilación de programas. Sin embargo, si especifica ADDR en una sentencia PROCESS (o CBL):

- En una compilación por lotes, sólo puede especificar ADDR para el primer programa. No puede cambiar el valor de la opción para programas posteriores en el lote.
- Debe utilizar la opción de 32 bits o 64 bits coincidente en el paso de enlace.

Si especifica opciones de compilador utilizando la opción -q del mandato cob2, puede abreviar ADDR(32) como 32 o ADDR(64) como 64. Por ejemplo:

```
cob2 -q64 prog64.cbl
```

Asignación de almacenamiento:

La asignación de almacenamiento para los siguientes tipos de datos COBOL depende del valor de la opción de compilador ADDR :

- USAGE POINTER (también los registros especiales ADDRESS OF , que implícitamente tiene este uso)
- USAGE PROCEDURE-POINTER
- USAGE FUNCTION-POINTER
- USAGE INDEX

Si ADDR(32) está en vigor, se asignan 4 bytes para cada elemento del programa que tenga uno de los usos listados anteriormente; si ADDR(64) está en vigor, se asignan 8 bytes para cada uno de los elementos.

Si se especifica la cláusula SYNCHRONIZED para un elemento de datos que tiene uno de los usos mostrados anteriormente, el elemento se alinea en un límite de palabra completa si ADDR(32) está en vigor, o en un límite de palabra doble si ADDR(64) está en vigor.

Registro especial de LENGTH OF :

Si ADDR(32) está en vigor, el registro especial LENGTH OF tiene esta definición implícita:

```
PICTURE 9(9) USAGE IS BINARY
```

Si ADDR(64) está en vigor, el registro especial LENGTH OF tiene esta definición implícita:

```
PICTURE 9(18) USAGE IS BINARY
```

Función intrínseca de LENGTH :

Si ADDR(32) está en vigor, el valor devuelto de la función intrínseca LENGTH es un entero de 9 dígitos. Si ADDR(64) está en vigor, el valor devuelto es un entero de 18 dígitos.

Requisitos y restricciones de programación:

- Todos los componentes de programa de una aplicación deben compilarse utilizando el mismo valor de la opción ADDR . No puede mezclar programas de 32 bits y programas de 64 bits en una aplicación.
- **Comunicación entre lenguajes:** En aplicaciones de varios lenguajes, los programas COBOL se pueden enlazar con C, C++ u otros lenguajes compilados y pueden interoperar con Java a través de JNI (Java Native Interface).
- **CICS:** los programas COBOL que se ejecutarán en el entorno TXSeries o CICS TX deben ser de 32 bits.

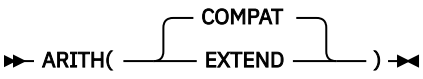
Tareas relacionadas
"Búsqueda de la longitud de los elementos de datos" en la página 112
"Compilación de programas" en la página 239
"Codificación de programas COBOL para ejecutarlos en CICS" en la página 409
"Llamada entre programas COBOL y C/C++" en la página 464

Referencias relacionadas
"Opciones cob2" en la página 248
"Opciones de compilador en conflicto" en la página 267

HARIT

ARITH afecta al número máximo de dígitos que puede codificar para elementos numéricos y al número de dígitos utilizados en los resultados intermedios de punto fijo.

Sintaxis de la opción ARITH



El valor predeterminado es: ARITH(COMPAT)

Las abreviaturas son: AR(C | E)

Cuando especifique ARITH(EXTEND):

- El número máximo de posiciones de dígitos que puede especificar en la cláusula PICTURE para los elementos de datos de decimal empaquetado, decimal externo y numéricos editados se eleva de 18 a 31.
- El número máximo de dígitos que puede especificar en un literal numérico de punto fijo se eleva de 18 a 31. Puede utilizar literales numéricos con gran precisión en cualquier lugar en el que se permitan actualmente los literales numéricos, incluyendo:
  - Operandos de sentencias PROCEDURE DIVISION
  - Cláusulas VALUE (para elementos de datos numéricos con precisión grande PICTURE)
  - Valores de nombre de condición (en elementos de datos numéricos con precisión grande PICTURE)
- El número máximo de dígitos que puede especificar en los argumentos en NUMVAL, NUMVAL-C y se eleva de 18 a 31.
- El valor máximo del argumento entero en la función FACTORIAL es 29.
- Los resultados intermedios en sentencias aritméticas utilizan la *modalidad ampliada*.

Cuando especifique ARITH(COMPAT):

- El número máximo de posiciones de dígitos en la cláusula PICTURE para elementos de datos de decimal empaquetado, decimal externo y editados numérica-editados es 18.
- El número máximo de dígitos en un literal numérico de punto fijo es 18.
- El número máximo de dígitos en los argumentos en NUMVAL, NUMVAL-C y es 18.
- El valor máximo del argumento de entero en la función FACTORIAL es 28.

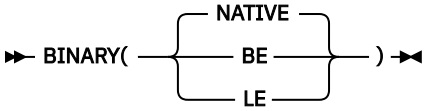
- Los resultados intermedios en sentencias aritméticas utilizan la *modalidad de compatibilidad*.

Conceptos relacionados
Apéndice B, “Resultados intermedios y precisión aritmética”, en la página 551

BINARIO

BINARY especifica el formato de representación de los elementos de datos binarios.

Sintaxis de la opción BINARY



El valor predeterminado es: `BINARY(NATIVE)`

Las abreviaturas son: Ninguna

Especifique `BINARY(NATIVE)` para utilizar el formato de representación binaria nativo de la plataforma. Para COBOL para Linux, este es el formato *little-endian* (dígito menos significativo en la dirección más baja).

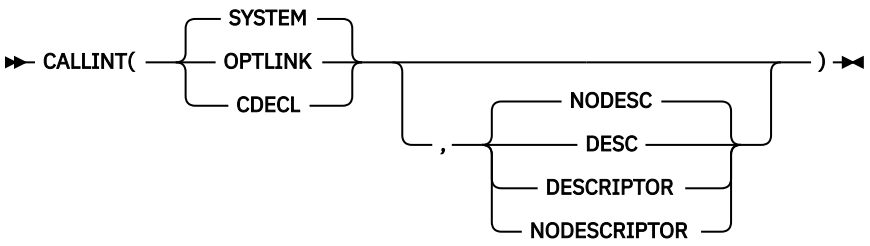
`BINARY(BE)` indica que los elementos de datos `BINARY`, `COMP`y `COMP-4` se representan de forma coherente con IBM Z, es decir, en formato *big-endian* (dígito más significativo en la dirección más baja).

`BINARY(LE)` indica que los elementos de datos `BINARY`, `COMP`y `COMP-4` se representan en formato *little-endian* (dígito menos significativo en la dirección más baja).

CALLINT

Utilice `CALLINT` para indicar el convenio de interfaz de llamada aplicable a las llamadas realizadas con la sentencia `CALL` y para indicar si se van a generar descriptores de argumento.

Sintaxis de la opción CALLINT



El valor predeterminado es: `CALLINT(SYSTEM,NODESC)`

Las abreviaturas son: Ninguna

Puede alterar temporalmente esta opción para sentencias `CALL` específicas utilizando la directiva de compilador `>>CALLINT`.

`CALLINT` tiene dos conjuntos de subopciones:

- Selección de un convenio de interfaz de llamada:

  SYSTEM
  La subopción SYSTEM especifica que el convenio de llamada es el del convenio de enlace de sistema estándar de la plataforma.

  SYSTEM es el único convenio de interfaz de llamada soportado en Linux.

OPTLINK
Si codifica la subopción OPTLINK , el compilador genera un mensajede diagnóstico de nivel I y se ignora toda la directiva (no sólo la primera palabra clave).

CDECL
Si codifica la subopción CDECL , el compilador genera un mensajede diagnóstico de nivel I y se ignora toda la directiva (no sólo la primera palabra clave).

- Especificación de si se van a generar los descriptores de argumento:

DESC
La subopción DESC especifica que se pasa un descriptor de argumento para cada argumento en una sentencia CALL . Para obtener más información sobre los descriptores de argumento, consulte las referencias relacionadas a continuación.

DESCRIPTOR
La subopción DESCRIPTOR es sinónimo de la subopción DESC .

NODESC
La subopción NODESC especifica que no se pasan descriptores de argumento para ningún argumento en una sentencia CALL .

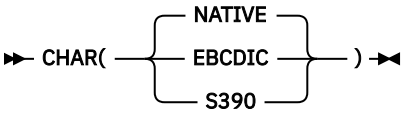
NODESCRIPTOR
La subopción NODESCRIPTOR es sinónimo de la subopción NODESC .

Referencias relacionadas
Capítulo 14, “Sentencias de direccionamiento de compilador”, en la página 309

char

CHAR afecta a la representación y al tratamiento en tiempo de ejecución de los elementos de datos USAGE DISPLAY y USAGE DISPLAY-1 .

Sintaxis de la opción CHAR



El valor predeterminado es: CHAR(NATIVE)

Las abreviaturas son: Ninguna

Especifique CHAR(NATIVE) para utilizar la representación de caracteres nativos (el formato nativo) de la plataforma. Para COBOL para Linux, el formato nativo se define mediante la página de códigos indicada por el entorno local en vigor en tiempo de ejecución. La página de códigos puede ser una página de códigos ASCII de un solo byte o una página de códigos Página de códigos multibyte basada en ASCII (UTF-8, EUC o ASCII DBCS).

CHAR(EBCDIC) y CHAR(S390) son sinónimos e indican que los elementos de datos DISPLAY y DISPLAY-1 están en la representación de caracteres de IBM Z, es decir, en EBCDIC.

Sin embargo, los elementos de datos DISPLAY y DISPLAY-1 definidos con la frase NATIVE en la cláusula USAGE no se ven afectados por la opción CHAR(EBCDIC) . Siempre se almacenan en el formato nativo de la plataforma.

La opción de compilador CHAR(EBCDIC) tiene los efectos siguientes en el proceso de tiempo de ejecución:

- Elementos **USAGE DISPLAY y USAGE DISPLAY-1 :** los caracteres de los elementos de datos que se describen con USAGE DISPLAY se tratan como formato EBCDIC de un solo byte. Los caracteres de los elementos de datos que se describen con USAGE DISPLAY-1 se tratan como formato EBCDIC DBCS. (En las viñetas siguientes, el término *EBCDIC* hace referencia al formato EBCDIC de un solo byte para USAGE DISPLAY y al formato EBCDIC DBCS para USAGE DISPLAY-1.)

  - Los datos codificados en el formato nativo se convierten al formato EBCDIC en ACCEPT desde el terminal.
  - Los datos EBCDIC se convierten al formato nativo en DISPLAY al terminal.
  - El contenido de literales alfanuméricos y literales DBCS se convierte al formato EBCDIC para su asignación a elementos de datos codificados en EBCDIC. Para ver las reglas sobre la comparación de datos de caracteres cuando la opción CHAR(EBCDIC) está en vigor, consulte la referencia relacionada a continuación sobre la opción COLLSEQ .
  - La edición se realiza con caracteres EBCDIC.
  - El relleno se realiza con espacios EBCDIC. Los elementos de grupo que se utilizan en operaciones alfanuméricas (como asignaciones y comparaciones) se rellenan con espacios EBCDIC de un solo byte independientemente de la definición de los elementos elementales dentro del grupo.
  - La constante figurativa SPACE o SPACES utilizada en una cláusula VALUE para una asignación a, o en una condición de relación con, un elemento USAGE DISPLAY se trata como un espacio EBCDIC de un solo byte (es decir, X'40').
  - La constante figurativa SPACE o SPACES utilizada en una cláusula VALUE para una asignación a, o en una condición de relación con, un elemento DISPLAY-1 se trata como un espacio EBCDIC DBCS (es decir, X'4040').
  - Las pruebas de clase se realizan basándose en rangos de valores EBCDIC.
- **USAGE DISPLAY elementos:**
  - El nombre de programa en CALL *identificador*, CANCEL *identificador*, o en una sentencia format-6 SET se convierte al formato nativo si el elemento de datos al que hace referencia el *identificador* está codificado en EBCDIC.
  - El nombre de archivo del elemento de datos al que hace referencia *nombre-datos* en ASSIGN USING *nombre-datos* se convierte al formato nativo si el elemento de datos se codifica en EBCDIC.
  - El nombre de archivo del registro especial SORT-CONTROL se convierte al formato nativo antes de pasarse a una función de clasificación o fusión. (SORT-CONTROL tiene la definición implícita USAGE DISPLAY.)
  - Los datos decimales con zona (cláusula PICTURE numérica con USAGE DISPLAY) y los datos de coma flotante display se tratan como formato EBCDIC. Por ejemplo, el valor de PIC S9 value "1" es X'F1' en lugar de X'31'.
- **Elementos de grupo:** Los elementos de grupo alfanuméricos se tratan de forma similar a los elementos de USAGE DISPLAY . (Tenga en cuenta que una cláusula USAGE para un grupo alfanumérico se aplica a los elementos elementales del grupo y no al propio grupo.)

Se presupone que los literales hexadecimales representan caracteres EBCDIC si los literales se asignan a, o se comparan con, datos de tipo carácter. Por ejemplo, X'C1' compara igual a un elemento alfanumérico que tiene el valor 'A'.

Las constantes figurativas HIGH-VALUE o HIGH-VALUES, LOW-VALUE o LOW-VALUES, SPACE o SPACES, ZERO o ZEROS, y QUOTE o QUOTES se tratan lógicamente como sus representaciones de caracteres EBCDIC para asignaciones o comparaciones con elementos de datos codificados en EBCDIC.

En las comparaciones entre elementos USAGE DISPLAY alfanuméricos, la secuencia de clasificación utilizada es la secuencia ordinal de los caracteres basada en sus valores binarios (hexadecimales) modificados por una secuencia de clasificación alternativa para caracteres de un solo byte, si se especifica.

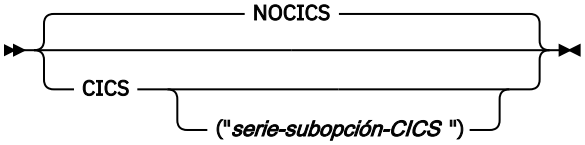
Tareas relacionadas
“Especificación de la página de códigos para datos de caracteres” en la página 214

Referencias relacionadas
“COLLSEQ” en la página 275
“La codificación de documentos XML” en la página 426
Apéndice A, “Consideraciones sobre el formato de datos de host de IBM Z”, en la página 549

CICS

La opción de compilador CICS habilita el conversor CICS integrado y le permite especificar subopciones CICS . Debe utilizar la opción CICS si el programa fuente COBOL contiene sentencias EXEC CICS y el programa no ha sido procesado por el conversor CICS independiente.

Sintaxis de la opción CICS



El valor predeterminado es: NOCICS

Las abreviaturas son: Ninguna

Utilice la opción CICS sólo para compilar programas CICS . Los programas compilados con la opción CICS no se ejecutarán en un entorno que no seaCICS .

Asegúrese de establecer las variables de entorno siguientes antes de compilar:

```
export NLSPATH=<CICS install dir>/msg/%L/%N:$NLSPATH
export LD_LIBRARY_PATH=<CICS install dir>/lib:$LD_LIBRARY_PATH
```

Si especifica la opción NOCICS , las sentencias CICS que se encuentren en el programa de origen se diagnosticarán y descartarán.

Utilice comillas o apóstrofos para delimitar la serie de subopciones CICS .

Puede utilizar la sintaxis mostrada anteriormente en la sentencia CBL o PROCESS . Si utiliza la opción CICS en el mandato cob2 o cob2_r , sólo se puede utilizar la comilla simple (') como delimitador de serie: -q"CICS('*options*')".

Puede particionar una serie de subopción CICS larga en varias series de subopción en varias sentencias CBL o PROCESS . Las subopciones CICS se concatenan en el orden en que aparecen. Por ejemplo, supongamos que el archivo de origen mypgm.cbl tiene el código siguiente:

```
cbl . . . CICS("string2") . . .
cbl . . . CICS("string3") . . .
```

Cuando emite el mandato cob2_r mypgm.cbl -q"CICS('string1')", el compilador pasa la siguiente serie de subopción al conversor CICS integrado:

```
"string1 string2 string3"
```

Las series concatenadas se delimitan con espacios únicos, tal como se muestra. Si se encuentran varias instancias de la misma subopción CICS , prevalecerá la última especificación de dicha subopción en la serie concatenada. El compilador limita el tamaño de la serie de subopción concatenada a 4 KB.

Conceptos relacionados
“Conversor integrado de CICS” en la página 414

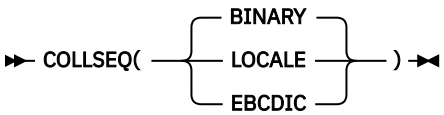
Tareas relacionadas
Capítulo 18, “Desarrollo de programas COBOL para CICS”, en la página 407

Referencias relacionadas
“Opciones de compilador en conflicto” en la página 267

COLLSEQ

COLLSEQ especifica el orden de clasificación para la comparación de operandos alfanuméricos y DBCS .

Sintaxis de la opción COLLSEQ



El valor predeterminado es: COLLSEQ(BINARY)

Las abreviaturas son: CS(L|E|BIN|B)

Puede especificar las siguientes subopciones para COLLSEQ:

- COLLSEQ(EBCDIC): utilice la secuencia de clasificación EBCDIC en lugar de la secuencia de clasificación ASCII.
- COLLSEQ(LOCALE): utilice la ordenación basada en el entorno local (coherente con los convenios culturales para la ordenación del entorno local).
- COLLSEQ(BIN): Utilice los valores hexadecimales de los caracteres; el valor de entorno local no tiene ningún efecto. Este valor proporcionará un mejor rendimiento en tiempo de ejecución.

Si utiliza la cláusula PROGRAM COLLATING SEQUENCE en el origen con un nombre de alfabeto de STANDARD-1, STANDARD-2o EBCDIC, la opción COLLSEQ se ignora para la comparación de operandos alfanuméricos. Si especifica PROGRAM COLLATING SEQUENCE is NATIVE, se aplica la opción COLLSEQ . De lo contrario, cuando el nombre de alfabeto especificado en la cláusula PROGRAM COLLATING SEQUENCE se define con literales, la secuencia de clasificación utilizada es la proporcionada por la opción COLLSEQ , modificada por la secuencia definida por el usuario dada por el nombre de alfabeto. (Para obtener detalles, consulte la referencia relacionada sobre la cláusula ALPHABET .)

La cláusula PROGRAM COLLATING SEQUENCE no tiene ningún efecto en las comparaciones DBCS.

La subopción anterior NATIVE está en desuso. Si especifica la subopción NATIVE , se asume COLLSEQ(LOCALE) .

La tabla siguiente resume la conversión y el orden de clasificación aplicables, basándose en los tipos de datos (ASCII o EBCDIC) utilizados en una comparación y la opción COLLSEQ en vigor cuando no se especifica la cláusula PROGRAM COLLATING SEQUENCE . Si se especifica, la especificación de origen tiene prioridad sobre la especificación de opción de compilador. La opción CHAR afecta a si los datos son ASCII o EBCDIC.

Tabla 31. ***Efecto del tipo de datos de comparación y del orden de clasificación en las comparaciones***

Comparandos	**COLLSEQ(BIN)**	**COLLSEQ(LOCALE)**	**COLLSEQ(EBCDIC)**
Ambos ASCII	No se realiza ninguna conversión. La comparación se basa en el valor binario (ASCII).	No se realiza ninguna conversión. La comparación se basa en el entorno local actual.	Ambos comparados se convierten a EBCDIC. La comparación se basa en el valor binario (EBCDIC).
ASCII mixto y EBCDIC	La comparación EBCDIC se convierte a ASCII. La comparación se basa en el valor binario (ASCII).	La comparación EBCDIC se convierte a ASCII. La comparación se basa en el entorno local actual.	La comparación ASCII se convierte a EBCDIC. La comparación se basa en el valor binario (EBCDIC).

*Tabla 31. **Efecto del tipo de datos de comparación y del orden de clasificación en las comparaciones** (continuación)*			
Comparandos	**COLLSEQ(BIN)**	**COLLSEQ(LOCALE)**	**COLLSEQ(EBCDIC)**
Ambos EBCDIC	No se realiza ninguna conversión. La comparación se basa en el valor binario (EBCDIC).	Las comparaciones se convierten a ASCII. La comparación se basa en el entorno local actual.	No se realiza ninguna conversión. La comparación se basa en el valor binario (EBCDIC).

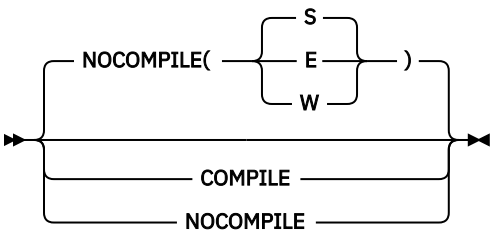
Tareas relacionadas
“Especificación del orden de clasificación” en la página 6
“Control de la secuencia de clasificación con un entorno local” en la página 219

Referencias relacionadas
“char” en la página 272
Cláusula ALPHABET (*COBOL for Linux en x86 Consulta de lenguaje*)

COMPILAR

Utilice la opción COMPILE sólo si desea forzar la compilación completa incluso en presencia de errores graves. Se generarán todos los diagnósticos y el código de objeto. No intente ejecutar el código de objeto si la compilación ha generado errores graves: los resultados podrían ser imprevisibles o podría producirse una terminación anómala.

Sintaxis de la opción COMPILE



El valor predeterminado es: NOCOMPILE(S)

Las abreviaturas son: C | NOC

Utilice NOCOMPILE sin ninguna subopción para solicitar una comprobación de sintaxis (sólo diagnósticos producidos, sin código de objeto). Si utiliza NOCOMPILE sin ninguna subopción, varias opciones de compilador no tendrán ningún efecto porque no se producirá ningún código de objeto, por ejemplo: LIST, OPTIMIZE, SSRANGEy TEST.

Utilice NOCOMPILE con la subopción W, Eo S para la compilación completa condicional. La compilación completa (diagnóstico y código de objeto) se detendrá cuando el compilador encuentre un error del nivel que especifique (o superior), y sólo continuará la comprobación de sintaxis.

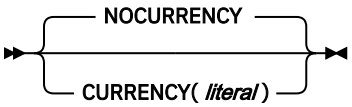
Tareas relacionadas
“Búsqueda de errores de codificación” en la página 323

Referencias relacionadas
“Mensajes y listados de errores detectados por el compilador” en la página 246

MONEDA

Puede utilizar la opción CURRENCY para proporcionar un símbolo de moneda predeterminado alternativo que se utilizará para un programa COBOL. (El símbolo de moneda predeterminado es el signo de dólar ($).)

Sintaxis de la opción CURRENCY



El valor predeterminado es: NOCURRENCY

Las abreviaturas son: CURR | NOCURR

NOCURRENCY especifica que no se utilizará ningún símbolo de moneda predeterminado alternativo.

Para cambiar el símbolo de moneda predeterminado, especifique CURRENCY(*literal*), donde *literal* es un literal alfanumérico COBOL válido (opcionalmente un literal hexadecimal) que representa un único carácter. El literal no debe estar en la lista siguiente:

- Dígitos cero (0) a nueve (9)
- Caracteres alfabéticos en mayúsculas A B C D E G N P R S V X Z o sus equivalentes en minúsculas
- El espacio
- Caracteres especiales * +-/,. ; () " =
- Una constante figurativa
- Un literal terminado en nulo
- Un literal DBCS
- Un literal nacional

Si el programa sólo procesa un tipo de moneda, puede utilizar la opción CURRENCY como alternativa a la cláusula CURRENCY SIGN para indicar el símbolo de moneda que utilizará en la cláusula PICTURE del programa. Si el programa procesa más de un tipo de moneda, debe utilizar la cláusula CURRENCY SIGN con la frase WITH PICTURE SYMBOL para especificar los distintos tipos de signo de moneda.

Si utiliza la opción CURRENCY y la cláusula CURRENCY SIGN en un programa, la opción CURRENCY se ignora. Los símbolos de moneda especificados en la cláusula o cláusulas CURRENCY SIGN se pueden utilizar en cláusulas PICTURE .

Cuando la opción NOCURRENCY está en vigor y omite la cláusula CURRENCY SIGN , el signo de dólar ($) se utiliza como símbolo PICTURE para el signo de moneda.

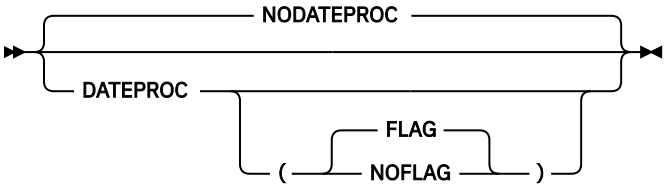
Delimitador: puede delimitar el literal de opción CURRENCY con comillas o apóstrofos, independientemente del valor de la opción de compilador APOST | QUOTE .

Tareas relacionadas
"Utilización de signos de moneda" en la página 56

DATEPROC

Utilice la opción DATEPROC para habilitar las extensiones de lenguaje millennium del compilador COBOL.

Sintaxis de la opción DATEPROC



El valor predeterminado es: NODATEPROC, o DATEPROC(FLAG) si sólo se especifica DATEPROC

Las abreviaturas son: DP | NODP

DATEPROC(FLAG)
Con DATEPROC(FLAG), las extensiones de lenguaje millennium están habilitadas y el compilador genera un mensaje de diagnóstico siempre que un elemento de lenguaje utilice o se vea afectado por las extensiones. El mensaje suele ser un mensaje de nivel de información o de nivel de aviso que identifica sentencias que implican un proceso sensible a la fecha. Se pueden generar mensajes adicionales que identifiquen errores o posibles incoherencias en las construcciones de fecha.

La producción de mensajes de diagnóstico, y su aspecto dentro o después del listado de origen, está sujeta al valor de la opción de compilador FLAG .

DATEPROC(NOFLAG)
Con DATEPROC(NOFLAG), las extensiones de lenguaje del milenio están en vigor, pero el compilador no produce ningún mensaje relacionado a menos que haya errores o incoherencias en el código fuente COBOL.

NODATEPROC
NODATEPROC indica que las extensiones no están habilitadas para esta unidad de compilación. Esta opción afecta a las construcciones de programa relacionadas con la fecha como se indica a continuación:

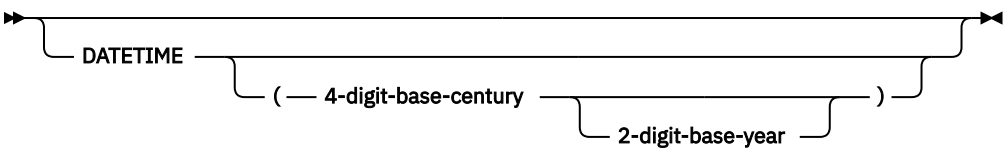
- La cláusula DATE FORMAT se comprueba con la sintaxis, pero no tiene ningún efecto en la ejecución del programa.
- Las funciones intrínsecas DATEVAL y UNDATE no tienen ningún efecto. Es decir, el valor devuelto por la función intrínseca es exactamente el mismo que el valor del argumento.
- La función intrínseca YEARWINDOW devuelve un valor de cero.

Referencias relacionadas
"BANDERA" en la página 284
"VENTANA" en la página 307

FECHA Y HORA

La opción DATETIME especifica la ventana de fecha que se utiliza para el algoritmo de ventana.

Sintaxis de la opción DATETIME



El valor predeterminado es: DATETIME(1900, 40)

Las abreviaturas son: Ninguna

4-digit-base-century
Este debe ser el primer argumento. Define el siglo base utilizado para el algoritmo de ventana. El valor predeterminado es 1900.

2-digit-base-year
Este debe ser el segundo argumento. Define el año base utilizado para el algoritmo de ventanas. El valor predeterminado es 40.

La opción predeterminada `DATETIME(1900, 40)` da como resultado una ventana de 100 años de 1940 a 2039. La especificación de `DATETIME(1900 70)` da como resultado una ventana de 100 años de 1970 a 2069.

Especificación de opción:

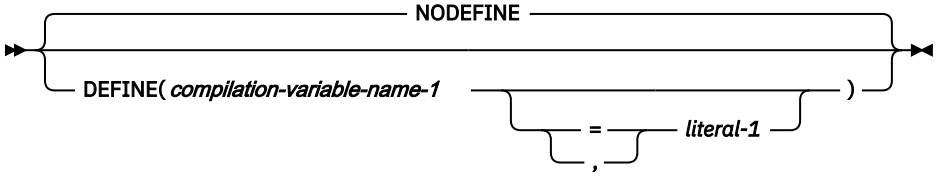
- Como opción en el mandato cob2 , los argumentos deben estar entre comillas simples; por ejemplo, DATETIME ('1900 40').
- Como sentencia de proceso, los argumentos se pueden escribir sin comillas; por ejemplo, DATETIME (1900 40).
- Si no se especifica la opción DATETIME o no se proporciona ningún argumento, tanto el siglo base como el año base toman sus valores predeterminados. El siglo base se puede especificar sin un argumento de año base siguiente, en cuyo caso el año base tomará su valor predeterminado. Si se especifica el año base, también se debe especificar el siglo base.

DEFECTO

Utilice la opción de compilador `DEFINE` para asignar un valor literal a una variable de compilación definida en el programa utilizando la directiva `DEFINE` con la frase `PARAMETER` . El valor literal proporcionado para la variable de compilación en la opción `DEFINE` a veces se conoce como un "valor de parámetro" para la variable de compilación correspondiente. Las variables de compilación se pueden utilizar dentro de cualquiera de las directivas de compilación condicionales, incluyendo `DEFINE`, `EVALUATE`y `IF`. Cuando una variable de compilación condicional aparece en una directiva de compilación condicional, se trata como una referencia simbólica al valor literal que representa actualmente.

La opción de compilador `DEFINE` proporciona una forma de asignar valores a variables de compilación de fuera del origen del programa. Si esto no es necesario, basta con utilizar la directiva `DEFINE` dentro del fuente del programa para definir variables de compilación.

Sintaxis de la opción DEFINE



El valor predeterminado es: `NODEFINE`

Las abreviaturas son: `DEF` | `NODEF`

compilation-variable-name-1
El nombre de una variable de compilación a la que se hace referencia en las directivas de compilación condicional del programa. Si no existe ninguna directiva `DEFINE` correspondiente con la frase `PARAMETER` para *compilation-variable-name-1* en el programa, se ignorará cualquier instancia de la opción de compilador `DEFINE` especificada para dicha variable de compilación. *compilation-variable-name-1* se forma de acuerdo con las reglas de una palabra definida por el usuario de nombre de datos, excepto que los caracteres DBCS no están permitidos en el nombre. Para obtener más detalles, Consulte *Palabras definidas por el usuario* en *COBOL for Linux en x86 Consulta de lenguaje*.

literal-1
El valor literal que *compilation-variable-name-1* representará simbólicamente en las directivas relacionadas con la compilación condicional en el programa. *literal-1* debe ser uno de los elementos siguientes:

- Un literal alfanumérico, que se puede especificar como un literal alfanumérico normal ('abcd ') o como un literal hexadecimal (x'F1F2F3'). Los literales nacionales, los literales DBCS y los literales alfanuméricos terminados en nulo (literales Z) no están soportados.
- Un literal entero.
- Un literal booleano (solo se da soporte a B '0' y B '1').

Si no se especifica *literal-1* , se asignará un valor de B '1' a la variable de compilación. Por ejemplo, si especifica:

```
>>define foo
```

A `foo` se le asignará el valor B '1'.

Nota: El compilador interpreta determinados caracteres de script de shell como se indica a continuación:

- Un signo igual (=) se interpreta como un paréntesis izquierdo, (
- Un signo de dos puntos (:) se interpreta como un paréntesis derecho,)
- Un subrayado (_) se interpreta como una comilla simple (')

Puede añadir un carácter de escape de barra inclinada invertida (\) para evitar la interpretación y, por lo tanto, para pasar caracteres en las series. Si desea que la barra inclinada invertida (\) se represente a sí misma (en lugar de como un carácter de escape), utilice la barra inclinada invertida doble (\\).

Por ejemplo, para utilizar la opción DEFINE para asignar el valor literal 1 a una variable de compilación VAR1, especifique la opción DEFINE de la siguiente manera:

```
DEFINE(VAR1=1)
```

Si VAR1 contiene un signo de igual, dos puntos o un subrayado que desea escapar de la interpretación del compilador, especifique la opción DEFINE de la siguiente manera:

```
DEFINE(VAR1\=1)
```

Se pueden especificar varias instancias de la opción `DEFINE` para definir un valor para varias variables de compilación diferentes. Si se define una única variable de compilación condicional más de una vez, se utilizará la última definición de la variable como valor de la variable de compilación condicional correspondiente. Si aparece `NODEFINE` después de instancias anteriores de la opción `DEFINE` , se cancelan las definiciones para todas las variables de compilación condicionales.

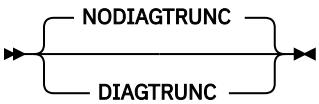
Cuando se especifican opciones `DEFINE` en sentencias CBL, sólo se pueden utilizar en el primer programa de un programa por lotes. Por lo tanto, si un archivo tiene varios programas COBOL, puede haber sentencias CBL con opciones `DEFINE` antes del primer programa, pero no los otros programas. Las opciones `DEFINE` especificadas para el primer programa (y las opciones `DEFINE` especificadas como opciones de mandato cob2) se aplican a todos los programas de un archivo.

Referencias relacionadas
Compilación condicional (*COBOL for Linux en x86 Consulta de lenguaje*)
DEFINE (*COBOL for Linux en x86 Consulta de lenguaje*)

DIAGTRUNC

`DIAGTRUNC` hace que el compilador emita un mensaje de diagnóstico severity-4 (Aviso) para sentencias `MOVE` que tienen receptores numéricos cuando el elemento de datos de recepción tiene menos posiciones de enteros que el literal o elemento de datos de envío. En las sentencias que tienen varios receptores, el mensaje se emite por separado para cada receptor que podría truncarse.

Sintaxis de la opción DIAGTRUNC



El valor predeterminado es: NODIAGTRUNC

Las abreviaturas son: DTR | NODTR

El mensaje de diagnóstico también se emite para movimientos implícitos asociados con sentencias como las siguientes:

- INITIALIZE
- READ . . . INTO
- RELEASE . . . FROM
- RETURN . . . INTO
- REWRITE . . . FROM
- WRITE . . . FROM

El mensaje de diagnóstico también se emite para traslados a receptores numéricos desde nombres de datos alfanuméricos o remitentes literales, excepto cuando se modifica el campo de envío como referencia.

No hay ningún mensaje de diagnóstico para los receptores COMP-5 , ni para los receptores binarios cuando se especifica la opción TRUNC(BIN) .

Conceptos relacionados
"Formatos para datos numéricos" en la página 39
"Modificadores de referencia" en la página 103

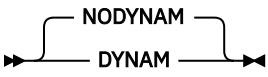
Referencias relacionadas
"TRUNC" en la página 302

DYNAM

Utilice DYNAM para hacer que los programas compilados por separado no anidados invocados a través de la sentencia CALL *literal* se carguen para CALLy se supriman para CANCEL, dinámicamente en tiempo de ejecución.

Las sentencias CALL *identifier* siempre dan como resultado una carga de tiempo de ejecución del programa de destino y no se ven afectadas por esta opción.

Sintaxis de la opción DYNAM



El valor predeterminado es: NODYNAM

Las abreviaturas son: DYN | NODYN

La condición para la frase ON EXCEPTION puede producirse para una sentencia CALL *literal* sólo si la opción DYNAM está en vigor.

Restricción: La opción de compilador DYNAM no debe utilizarse para programas que contengan sentencias EXEC CICS o EXEC SQL.

Con NODYNAM, el nombre de programa de destino se resuelve a través del enlazador.

Con la opción DYNAM , la sentencia siguiente:

```
CALL "myprogram" . . .
```

tiene el mismo comportamiento que estas sentencias:

```
MOVE "myprogram" to id-1
CALL id-1 ...
```

Conceptos relacionados
“Identificador CALL y literal CALL” en la página 462
CALLINTERFACE (*COBOL for Linux en x86 Consulta de lenguaje*)

Referencias relacionadas
“Opciones de compilador en conflicto” en la página 267

salida

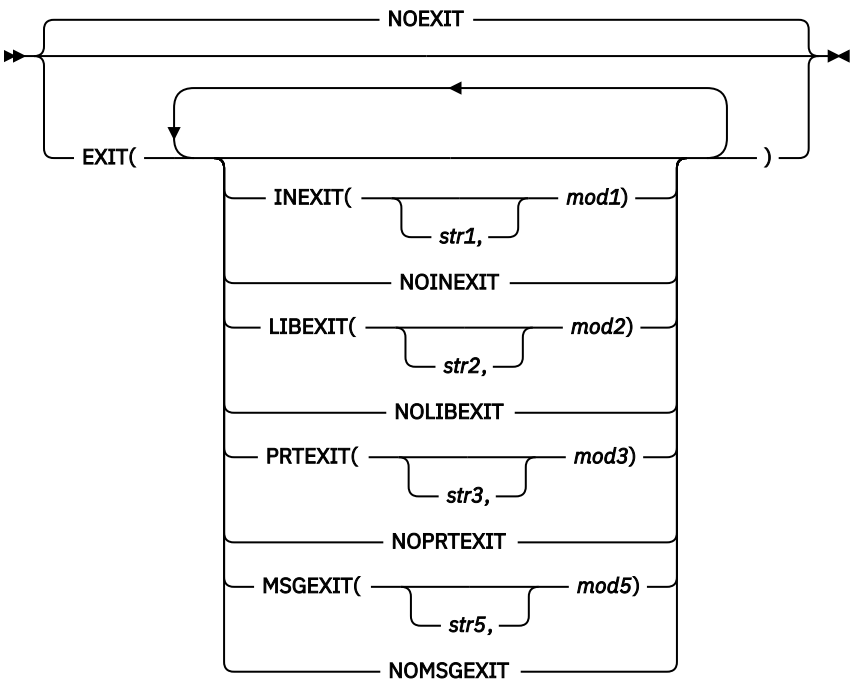
Utilice la opción EXIT para proporcionar módulos proporcionados por el usuario en lugar de varias funciones del compilador.

Para la entrada de compilador, utilice la subopción INEXIT para proporcionar un módulo en lugar de SYSIN (entrada de compilador primario) y utilice la subopción LIBEXIT para proporcionar un módulo en lugar de SYSLIB (entrada de biblioteca de copia). Para la salida del compilador, utilice la subopción PRTEXIT para proporcionar un módulo en lugar de SYSPRINT (el archivo de listado del compilador).

Para personalizar los mensajes del compilador (cambiar su gravedad o suprimirlos, incluida la conversión de mensajes FIPS (FLAGSTD) en mensajes de diagnóstico a los que asigna una gravedad), utilice la subopción MSGEXIT . El módulo que proporcione para personalizar los mensajes se llamará cada vez que el compilador emita un mensaje de diagnóstico o un mensaje FIPS.

Al crear el módulo de salida, asegúrese de que el módulo esté enlazado como un módulo de biblioteca compartida antes de ejecutarlo con el compilador COBOL. Los módulos de salida se invocan con el convenio de enlace del sistema de la plataforma.

Sintaxis de la opción EXIT



El valor predeterminado es: NOEXIT

Las abreviaturas son: NOEX | EX(INX | NOINX, LIBX | NOLIBX, PRTX | NOPRTX, ADX | NOADX, MSGX | NOMSGX)

Especificación de opción: No puede especificar la opción EXIT en una sentencia PROCESS (o CBL). Sólo puede especificarlo de una de las maneras siguientes:

- Como opción en el mandato cob2
- En la variable de entorno COBOPT

Puede especificar las subopciones en cualquier orden y puede separarlas mediante comas o espacios. Si especifica tanto el formato positivo como el negativo de una subopción, el formato especificado entra en vigor en último lugar. Si especifica la misma subopción más de una vez, la última especificada entrará en vigor.

Si especifica la opción EXIT sin proporcionar al menos una subopción (es decir, especifica EXIT()), NOEXIT estará en vigor.

INEXIT([*'str1'*,]*mod1*)
El compilador lee el código fuente de un módulo de carga proporcionado por el usuario (donde *mod1* es el nombre del módulo) en lugar de SYSIN.

LIBEXIT([*'str2'*,]*mod2*)
El compilador obtiene libros de copias de un módulo de carga proporcionado por el usuario (donde *mod2* es el nombre del módulo) en lugar de *nombre-biblioteca* o SYSLIB. Para su uso con sentencias COPY o BASIS .

PRTEXIT([*'str3'*,]*mod3*)
El compilador pasa la salida destinada a la impresora al módulo de carga proporcionado por el usuario (donde *mod3* es el nombre del módulo) en lugar de SYSPRINT.

MSGEXIT(['*str5*',]*mod5*)
El compilador pasa el número de mensaje y pasa la gravedad predeterminada de un mensaje de diagnóstico de compilador, o la categoría (como código numérico) de un mensaje de compilador FIPS, al módulo de carga proporcionado por el usuario (donde *mod5* es el nombre del módulo).

Los nombres *mod1*, *mod2*, *mod3*, *mod4*y *mod5* pueden hacer referencia al mismo módulo.

Las subopciones *str1*, *str2*, *str3*, *str4*y *str5* son series de caracteres que se pasan al módulo de carga. Estas series son opcionales. Pueden tener hasta 64 caracteres de longitud, y debe encerrarlos entre un par de apóstrofos (''). Puede utilizar cualquier carácter en las series, pero los apóstrofos incluidos deben doblarse ('). Los caracteres en minúsculas se convierten a mayúsculas.

Formatos de serie de caracteres: Si uno de *str1*, *str2*, Se especifica *str3*, *str4*o *str5* , esa serie se pasa al módulo de salida de usuario adecuado en el formato siguiente, donde LL es una media palabra (en un límite de media palabra) que contiene la longitud de la serie.

Arrendador	Serie

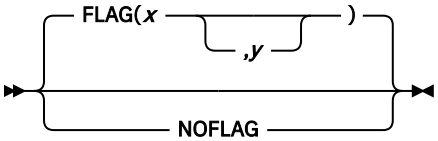
"Ejemplo: salida de usuario MSGEXIT" en la página 619

Referencias relacionadas
"FLAGSTD" en la página 285
Apéndice E, "Opción de compilador EXIT", en la página 611

BANDERA

Utilice FLAG(*x*) para generar mensajes de diagnóstico al final del listado de origen para los errores de un nivel de gravedad *x* o superior.

Sintaxis de la opción FLAG



El valor predeterminado es: FLAG(I,I)

Las abreviaturas son: F | NOF

x y *y* pueden ser I, W, E, So U.

Utilice FLAG(*x*,*y*) para generar mensajes de diagnóstico para errores de nivel de gravedad *x* o superior al final del listado de origen, con mensajes de error de gravedad *y* o superior que se deben incluir directamente en el listado de origen. La gravedad codificada para *y* no debe ser inferior a la gravedad codificada para *x*. Para utilizar FLAG(*x*,*y*), también debe especificar la opción de compilador SOURCE .

Los mensajes de error en el listado fuente se desactivan al incluir el número de sentencia en una flecha que apunta al código de mensaje. El código de mensaje va seguido del texto del mensaje. Por ejemplo:

```
  000413     MOVE CORR WS-DATE TO HEADER-DATE
==000413==>    IGYPS2121-S     " WS-DATE " was not defined as a data-name.  . . .
```

Cuando FLAG(*x*,*y*) está en vigor, la mayoría de los mensajes de gravedad *y* y superiores se incorporan en el listado después de la línea que ha causado el mensaje. Los mensajes con el prefijo IGYCB nunca se incorporarán en el origen. (Consulte la referencia relacionada a continuación para obtener información sobre los mensajes para excepciones.)

Utilice NOFLAG para suprimir el etiquetado de errores. NOFLAG no suprime los mensajes de error para las opciones de compilador.

Mensajes incorporados

- No se recomienda incluir mensajes de nivel U. Se acepta la especificación de mensajes de nivel U incorporados, pero no genera ningún mensaje en el origen.
- La opción FLAG no afecta a los mensajes de diagnóstico que se generan antes de que se procesen las opciones del compilador.
- Los mensajes de diagnóstico que se generan durante el proceso de opciones de compilador, sentencias CBL o PROCESS , o sentencias BASIS, COPYo REPLACE no se incorporan en el listado fuente. Todos estos mensajes aparecen al principio de la salida del compilador.
- Los mensajes de diagnóstico con el prefijo IGYCB no están incorporados en el listado de origen. Todos estos mensajes aparecen al final de la salida del compilador, independientemente del valor de la opción FLAG .
- Los mensajes producidos durante el proceso de la sentencia *CONTROL o *CBL no se incorporan en el listado fuente.

Referencias relacionadas
“Mensajes y listados de errores detectados por el compilador” en la página 246

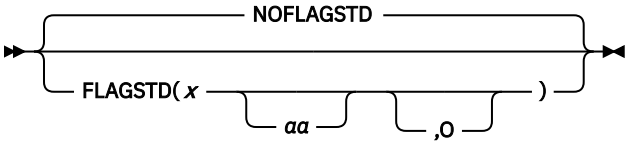
FLAGSTD

Utilice FLAGSTD para especificar el nivel o subconjunto del 85 COBOL Estándar que se debe considerar conforme y para obtener mensajes informativos sobre los elementos de 85 COBOL Estándar que se incluyen en el programa.

Puede especificar cualquiera de los elementos siguientes para la marcación:

- Un subconjunto seleccionado de FIPS (Federal Information Processing Standard) COBOL
- Cualquiera de los módulos opcionales
- Elementos de lenguaje obsoletos
- Cualquier combinación de subconjunto y módulos opcionales
- Cualquier combinación de subconjunto y elementos obsoletos
- IBM (se marcan en cualquier momento en que se especifica FLAGSTD y se identifican como "no conformes no estándar")

Sintaxis de la opción FLAGSTD



El valor predeterminado es: NOFLAGSTD

Las abreviaturas son: Ninguna

x especifica el subconjunto de 85 COBOL Estándar que se debe considerar conforme:

M
Los elementos de lenguaje que no son del subconjunto mínimo se marcarán como "estándar no conforme".

I
Los elementos de lenguaje que no son del mínimo o del subconjunto intermedio se marcarán como "estándar no conforme".

H
Se está utilizando el subconjunto alto y el subconjunto no marcará los elementos. Los elementos que son extensiones de IBM se marcarán como "Estándar no conforme, extensión IBM ."

yy especifica, mediante un único carácter o combinación de dos cualquiera, los módulos opcionales que se incluirán en el subconjunto:

D
Los elementos del nivel 1 del módulo de depuración no están marcados como "estándar no conforme".

N
Los elementos del nivel 1 del módulo de segmentación no se marcan como "estándar no conforme".

S
Los elementos del nivel 2 del módulo de segmentación no se marcan como "estándar no conforme".

Si se especifica S , se incluye N (N es un subconjunto de S).

0 (la letra) especifica que los elementos de lenguaje obsoletos se marcan como "obsoletos".

Los mensajes informativos aparecen en el listado del programa fuente e identifican:

- El elemento como "obsoleto", "no conforme estándar" o "no conforme estándar" (un elemento de lenguaje que es obsoleto y no conforme sólo se marca como obsoleto)
- La cláusula, sentencia o cabecera que contiene el elemento
- La línea del programa fuente y la ubicación inicial de la cláusula, sentencia o cabecera que contiene el elemento
- El subconjunto o módulo opcional al que pertenece el elemento

FLAGSTD requiere el conjunto estándar de palabras reservadas.

En el ejemplo siguiente, el número de línea y la columna donde se ha producido una cláusula, sentencia o cabecera marcada se muestran con el código de mensaje y el texto asociados. Después de eso es un resumen del número total de elementos marcados y su tipo.

```
   LINE.COL CODE       FIPS MESSAGE TEXT

            IGYDS8211  Comment lines before "IDENTIFICATION DIVISION":
                       nonconforming nonstandard, IBM extension to
                       ANS/ISO 1985.

      11.14 IGYDS8111  "GLOBAL clause":  nonconforming standard, ANS/ISO
                       1985 high subset.

      59.12 IGYPS8169  "USE FOR DEBUGGING statement":  obsolete element
                       in ANS/ISO 1985.


 FIPS MESSAGES TOTAL             STANDARD      NONSTANDARD      OBSOLETE

                 3                   1              1              1
```

Puede convertir mensajes informativos FIPS en mensajes de diagnóstico, y puede suprimir mensajes FIPS, utilizando la subopción MSGEXIT de la opción de compilador EXIT . Para obtener detalles, consulte la referencia relacionada sobre el proceso de MSGEXIT y consulte la tarea relacionada.

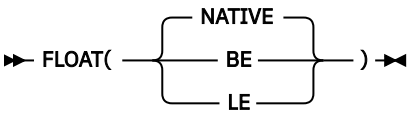
Tareas relacionadas
“Personalización de gravedades de mensajes del compilador” en la página 617

Referencias relacionadas
“Proceso de MSGEXIT” en la página 616

FLOAT

Float especifica el formato de representación de los elementos de datos de coma flotante.

Sintaxis de la opción FLOAT



El valor predeterminado es: `FLOAT(NATIVE)`

Las abreviaturas son: Ninguna

Especifique `FLOAT(NATIVE)` para utilizar el formato de representación de coma flotante nativo de la plataforma. Para COBOL para Linux, este es el formato *little-endian* (dígito menos significativo en la dirección más baja).

`FLOAT(BE)` indica que los elementos de datos `COMP-1` y `COMP-2` se representan de forma coherente con IBM Z, es decir, en formato *big-endian* (dígito más significativo en la dirección más baja).

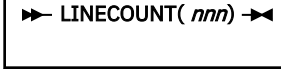
`FLOAT(LE)` indica que los elementos de datos `COMP-1` y `COMP-2` se representan en formato *little-endian* (dígito menos significativo en la dirección más baja).

Referencias relacionadas
Apéndice A, “Consideraciones sobre el formato de datos de host de IBM Z”, en la página 549

LINECOUNT

Utilice `LINECOUNT(nnn)` para especificar el número de líneas que deben imprimirse en cada página del listado de compilación, o utilice `LINECOUNT(0)` para suprimir la paginación.

Sintaxis de la opción LINECOUNT



El valor predeterminado es: `LINECOUNT(60)`

Las abreviaturas son: `LC`

nnn debe ser un entero entre 10 y 255, o 0.

Si especifica `LINECOUNT(0)`, no se generan expulsiones de página en el listado de compilación.

El compilador utiliza tres líneas de *nnn* para títulos. Por ejemplo, si especifica `LINECOUNT(60)`, se imprimen 57 líneas de código fuente en cada página del listado de salida.

lista

Utilice la opción de compilador `LIST` para generar un listado de la expansión del lenguaje ensamblador del código fuente.

Sintaxis de la opción LIST

El valor predeterminado es: `NOLIST`

Las abreviaturas son: Ninguna

Las sentencias *CONTROL (o *CBL) LIST o NOLIST que codifique en PROCEDURE DIVISION no tienen ningún efecto. Se tratan como comentarios.

El listado de assembler se graba en un archivo que tiene el mismo nombre que el programa fuente pero tiene el sufijo .wlist.

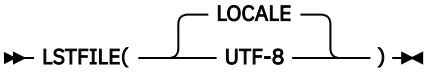
Tareas relacionadas
"Obtención de listados" en la página 382

Referencias relacionadas
*CONTROL (*CBL) (*COBOL for Linux en x86 Consulta de lenguaje*)

LSTFILE

Especifique LSTFILE(LOCALE) para que el listado de compilador generado se codifique en la página de códigos especificada por el entorno local en vigor. Especifique LSTFILE(UTF-8) para que el listado de compilador generado se codifique en UTF-8.

Sintaxis de la opción LSTFILE



El valor predeterminado es: LSTFILE(LOCALE)

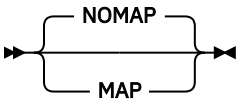
Las abreviaturas son: LST

Referencias relacionadas
Capítulo 11, "Establecimiento del entorno local", en la página 213

MAP

Utilice MAP para generar un listado de los elementos definidos en DATA DIVISION.

Sintaxis de la opción MAP



El valor predeterminado es: NOMAP

Las abreviaturas son: Ninguna

La salida incluye los elementos siguientes:

- DATA DIVISION MAP
- Correlación de estructura de programa anidada y atributos de programa
- Tamaño de WORKING-STORAGE y LOCAL-STORAGE del programa

Si desea limitar la salida de MAP , utilice sentencias *CONTROL MAP o NOMAP en DATA DIVISION. Las sentencias fuente que siguen a *CONTROL NOMAP no se incluyen en el listado hasta que una sentencia *CONTROL MAP vuelve a conmutar la salida al formato MAP normal. Por ejemplo:

```
*CONTROL NOMAP            *CBL NOMAP
    01  A                     01  A
    02  B                     02  B
*CONTROL MAP              *CBL MAP
```

Cuando la opción MAP está en vigor, también obtiene un informe MAP incorporado en el listado de código fuente. La información condensada de MAP se muestra a la derecha de las definiciones de nombre de datos en WORKING-STORAGE SECTION, FILE SECTION, LOCAL-STORAGE SECTIONy LINKAGE SECTION de DATA DIVISION. Cuando tanto los datos de XREF como un resumen de MAP incorporado están en la misma línea, el resumen de MAP incorporado se lista en primer lugar.

"Ejemplo: salida MAP" en la página 387

Conceptos relacionados
Capítulo 16, "depuración", en la página 319

Tareas relacionadas
"Obtención de listados" en la página 382

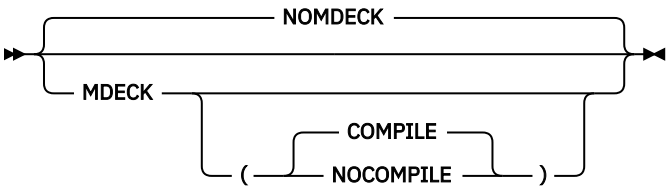
Referencias relacionadas
*CONTROL (*CBL) (*COBOL for Linux en x86 Consulta de lenguaje*)

MDECK

La opción de compilador MDECK especifica que una copia del origen de entrada actualizado después del proceso de biblioteca (es decir, el resultado de COPY, BASIS, REPLACE, y EXEC SQL INCLUDE) se graban en un archivo.

La salida MDECK se graba en el directorio actual en un archivo que tiene el mismo nombre que el archivo de origen COBOL y un sufijo de .dek.

Sintaxis de la opción MDECK



El valor predeterminado es: NOMDECK

Las abreviaturas son: NOMD | MD | MD(C | NOC)

Especificación de opción:

No puede especificar la opción MDECK en una sentencia PROCESS (o CBL). Sólo puede especificarlo de una de las maneras siguientes:

- Como opción en el mandato cob2
- En la variable de entorno **COBOPT**
- En el atributo compopts del archivo de configuración (.cfg)

Subopciones:

- Cuando MDECK(COMPILE) está en vigor, la compilación continúa normalmente después de que se hayan completado el proceso de biblioteca y la generación del archivo de salida MDECK , sujeto al valor de la opción COMPILE | NOCOMPILE.
- Cuando MDECK(NOCOMPILE) está en vigor, la compilación termina después de que se haya completado el comprobación de sintaxis y de que se haya grabado el archivo de programa fuente expandido. El compilador no realiza generación de código independientemente del valor de la opción COMPILE.

Si especifica MDECK sin subopción, MDECK(COMPILE) está implícito.

Contenido del archivo de salida MDECK :

Si utiliza la opción MDECK con programas que contienen EXEC CICS o EXEC SQL , estas sentencias EXEC se incluyen en la salida MDECK tal cual. Sin embargo, si compila utilizando la opción SQL , las sentencias EXEC SQL INCLUDE correspondientes se expanden en la salida MDECK .

Las imágenes de tarjeta CBL, PROCESS, *CONTROLy *CBL se pasan al archivo de salida MDECK en las ubicaciones adecuadas.

Para una compilación por lotes (varios programas fuente COBOL en un único archivo de entrada), se crea un único archivo de salida MDECK que contiene el origen expandido completo.

Cualquier proceso de opción de compilador SEQUENCE se refleja en el archivo MDECK .

Las sentencias COPY se incluyen en el archivo MDECK como comentarios.

Referencias relacionadas
“Stanzas en el archivo de configuración” en la página 244
“Opciones de compilador en conflicto” en la página 267
Capítulo 14, “Sentencias de direccionamiento de compilador”, en la página 309

NCOLLSEQ

NCOLLSEQ especifica el orden de clasificación para la comparación de los operandos nacionales de clase .

Sintaxis de la opción NCOLLSEQ

El valor predeterminado es: NCOLLSEQ(BINARY)

Las abreviaturas son: NCS(L|BIN|B)

NCOLLSEQ(BIN) utiliza los valores hexadecimales de los pares de caracteres.

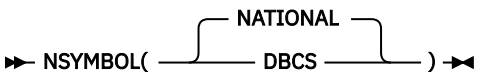
NCOLLSEQ(LOCALE) utiliza el algoritmo para el orden de clasificación que está asociado con el valor de entorno local que está en vigor.

Tareas relacionadas
“Comparación de dos operandos nacionales de clase” en la página 206
“Control de la secuencia de clasificación con un entorno local” en la página 219

SÍMBOLO

La opción NSYMBOL controla la interpretación del símbolo N utilizado en literales y cláusulas PICTURE , indicando si se presupone el proceso nacional o DBCS.

Sintaxis de la opción NSYMBOL



El valor predeterminado es: NSYMBOL(NATIONAL)

Las abreviaturas son: NS(NAT | DBCS)

Con NSYMBOL(NATIONAL):

- Los elementos de datos definidos con una cláusula PICTURE que consta sólo del símbolo N sin la cláusula USAGE se tratan como si se hubiera especificado la cláusula USAGE NATIONAL .
- Los literales con el formato N". . ." o N'. . .' se tratan como literales nacionales.

Con NSYMBOL(DBCS):

- Los elementos de datos definidos con una cláusula PICTURE que consta sólo del símbolo N sin la cláusula USAGE se tratan como si se hubiera especificado la cláusula USAGE DISPLAY-1 .
- Los literales con el formato N". . ." o N'. . .' se tratan como literales DBCS.

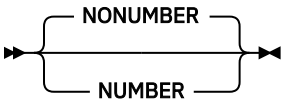
La opción NSYMBOL(DBCS) proporciona compatibilidad con releases anteriores de IBM COBOL, y la opción NSYMBOL(NATIONAL) hace que el manejo de los elementos de lenguaje anteriores sea coherente con el Estándar COBOL 2002 en este sentido.

NSYMBOL(NATIONAL) se recomienda para las aplicaciones que utilizan datos Unicode.

NÚMERO

Utilice la opción de compilador NUMBER si tiene números de línea en el código fuente y desea que estos números se utilicen en los mensajes de error y en los listados SOURCE, MAP, LISTy XREF .

Sintaxis de la opción NUMBER



El valor predeterminado es: NONUMBER

Las abreviaturas son: NUM | NONUM

Si solicita NUMBER, el compilador comprueba las columnas 1 a 6 para asegurarse de que sólo contienen números y que los números están en secuencia de clasificación numérica. (Por el contrario, SEQUENCE comprueba los caracteres de estas columnas de acuerdo con la secuencia de clasificación EBCDIC.) Cuando se encuentra que un número de línea está fuera de secuencia, el compilador le asigna un número de línea con un valor uno mayor que el número de línea de la sentencia anterior. El compilador señala el nuevo valor con dos asteriscos e incluye en el listado un mensaje que indica un error fuera de secuencia. La comprobación de secuencia continúa con la siguiente sentencia, basándose en el valor recién asignado de la línea anterior.

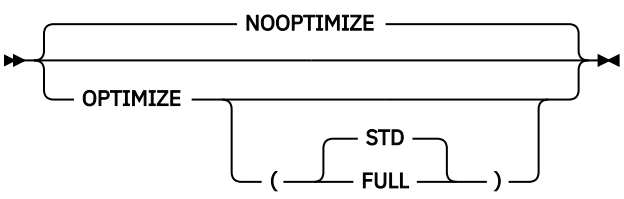
Si utiliza sentencias COPY y NUMBER está en vigor, asegúrese de que los números de línea del programa de origen y los números de línea del libro de copias estén coordinados.

Utilice NONUMBER si no tiene números de línea en el código fuente, o si desea que el compilador ignore los números de línea que tiene en el código fuente. Con NONUMBER en vigor, el compilador genera números de línea para las sentencias de origen y utiliza estos números como referencias en los listados.

OPTIMIZAR

Utilice OPTIMIZE para reducir el tiempo de ejecución del programa objeto. La optimización también puede reducir la cantidad de almacenamiento que utiliza el programa objeto. Las optimizaciones realizadas incluyen la propagación de constantes, la planificación de instrucciones y la eliminación de cálculos cuyos resultados nunca se utilizan.

Sintaxis de la opción OPTIMIZE



El valor predeterminado es: NOOPTIMIZE

Las abreviaturas son: OPT | NOOPT

Si se especifica OPTIMIZE sin ninguna subopción, OPTIMIZE(STD) está en vigor.

La subopción FULL solicita que, además de las optimizaciones realizadas con OPT(STD), el compilador descarte los elementos de datos no referenciados del DATA DIVISION y suprima la generación de código para inicializar estos elementos de datos en los valores de sus cláusulas VALUE . Cuando OPT(FULL) está en vigor, se descartan todos los elementos level-77 no referenciados y los elementos level-01 elementales. Además, los elementos de grupo level-01 se descartan si no se hace referencia a ninguno de sus elementos subordinados. Los elementos suprimidos se muestran en el listado. Si la opción MAP está en vigor, un BL número de XXXXX en la información de correlación de datos indica que el elemento de datos se ha descartado.

Recomendación: Utilice OPTIMIZE(FULL) para las aplicaciones de base de datos. Puede mejorar mucho el rendimiento, ya que se eliminan las constantes no utilizadas incluidas por las sentencias COPY asociadas. Sin embargo, si la aplicación de base de datos depende de elementos de datos no utilizados, consulte las recomendaciones siguientes.

Elementos de datos no utilizados: No utilice OPT(FULL) si los programas dependen de la utilización de elementos de datos no utilizados. En el pasado, esto se hacía comúnmente de dos maneras:

- Una técnica utilizada a veces en programas OS/VS COBOL antiguos era colocar una tabla no referenciada después de una tabla referenciada y utilizar subíndices fuera de rango en la primera tabla para acceder a la segunda tabla. Para determinar si los programas utilizan esta técnica, utilice la opción de compilador SSRANGE con la opción de tiempo de ejecución CHECK(ON). Para solucionar este problema, utilice la capacidad de COBOL más reciente para codificar tablas grandes y utilice sólo una tabla.
- Coloque elementos de datos llamativos en WORKING-STORAGE SECTION para identificar el principio y el final de los datos del programa o para marcar una copia de un programa para una herramienta de biblioteca que utiliza los datos para identificar la versión de un programa. Para resolver este problema, inicialice estos elementos con sentencias PROCEDURE DIVISION en lugar de cláusulas VALUE . Con este método, el compilador tendrá en cuenta estos elementos utilizados y no los suprimirá.

La opción OPTIMIZE está desactivada en el caso de un error de nivel grave o superior.

Conceptos relacionados
"Optimización" en la página 533

Referencias relacionadas
"Opciones de compilador en conflicto" en la página 267

PGMNAME

La opción PGMNAME controla el manejo de nombres de programa y nombres de punto de entrada.

Sintaxis de la opción GMNAME

El valor predeterminado es: PGMNAME(UPPER)

Las abreviaturas son: PGMN(LU|LM)

También se da soporte a la compatibilidad con COBOL for OS/390 & VM, LONGMIXED y LONGUPPER .

LONGUPPER se puede abreviar como UPPER, LUo U. LONGMIXED se puede abreviar como MIXED, LMo M.

COMPAT: si especifica PGMNAME(COMPAT), se establecerá PGMNAME(UPPER) y recibirá un mensaje de aviso.

PGMNAME controla el manejo de nombres utilizados en los contextos siguientes:

- Nombres de programa definidos en el párrafo PROGRAM-ID

- Nombres de punto de entrada de programa en la sentencia ENTRY
- Referencias de nombre de programa en:
  - Sentencias CALL que hacen referencia a programas anidados, programas enlazados estáticamente o Bibliotecas compartidas
  - Sentencias SET *procedure-pointer* o *function-pointer* que hacen referencia a programas enlazados estáticamente o Bibliotecas compartidas
  - Sentencias CANCEL que hacen referencia a programas anidados

PGMNAME (UPPER)

Con PGMNAME(UPPER), los nombres de programa especificados en el párrafo PROGRAM-ID como palabras definidas por el usuario COBOL deben seguir las reglas normales de COBOL para formar una palabra definida por el usuario:

- El nombre de programa puede tener una longitud máxima de 30 caracteres.
- Todos los caracteres utilizados en el nombre deben ser alfabéticos, dígitos, el guión, o el subrayado.
- Al menos un carácter debe ser alfabético.
- El guión no se puede utilizar como primer o último carácter.
- El carácter de subrayado no se puede utilizar como primer carácter.

Cuando se especifica un nombre de programa como un literal, en una definición o una referencia, entonces:

- El nombre de programa puede tener hasta 160 caracteres de longitud.
- Todos los caracteres utilizados en el nombre deben ser alfabéticos, dígitos, el guión, o el subrayado.
- Al menos un carácter debe ser alfabético.
- El guión no se puede utilizar como primer o último carácter.
- El subrayado se puede utilizar en cualquier posición.

Los nombres de programa externos se procesan con caracteres alfabéticos en mayúsculas.

PGMNAME (MIXTO)

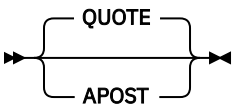
Con PGMNAME(MIXED), los nombres de programa se procesan tal cual, sin truncamiento, conversión o conversión a mayúsculas.

Con PGMNAME(MIXED), todas las definiciones de nombre de programa deben especificarse utilizando el formato literal del nombre de programa en el párrafo PROGRAM-ID o la sentencia ENTRY .

APOST/CITA

Utilice APOST si desea que la constante figurativa [ALL] QUOTE o [ALL] QUOTES represente uno o más caracteres de apóstrofo ('). Utilice QUOTE si desea que la constante figurativa [ALL] QUOTE o [ALL] QUOTES represente una o más comillas (") .

Sintaxis de la opción APOST/QUOTE



El valor predeterminado es: QUOTE

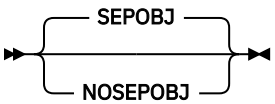
Las abreviaturas son: Q | APOST

Delimitadores: puede utilizar comillas (") o apóstrofos (') como delimitadores literales independientemente de si la opción APOST o QUOTE está en vigor. El carácter delimitador utilizado como delimitador de apertura para un literal debe utilizarse como delimitador de cierre para dicho literal.

SEPOBJ

SEPOBJ especifica si cada uno de los programas COBOL más externos de una compilación por lotes se va a generar como un archivo de objeto independiente en lugar de como un único archivo de objeto.

Sintaxis de la opción SEPOBJ



El valor predeterminado es: SEPOBJ

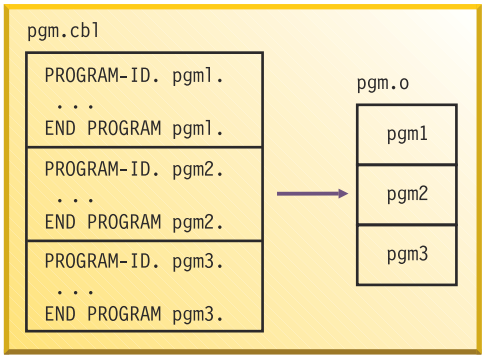
Las abreviaturas son: Ninguna

Compilación por lotes

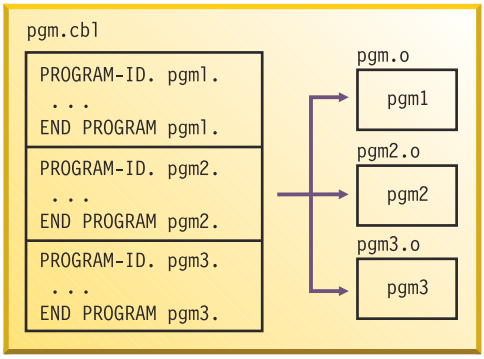
Cuando se compilan varios programas externos (programas no anidados) con una sola invocación por lotes del compilador, el número de archivos producidos para la salida de programa objeto de la compilación por lotes depende de la opción de compilador SEPOBJ.

Supongamos que el archivo de origen COBOL pgm.cbl contiene tres programas COBOL más externos denominados pgm1, pgm2y pgm3. Las figuras siguientes ilustran si la salida del programa objeto se genera como un archivo (con NOSEPOBJ) o tres archivos (con SEPOBJ).

Compilación por lotes con NOSEPOBJ



Compilación por lotes con SEPOBJ



Notas de uso

- La opción SEPOBJ es necesaria para ajustarse a 85 COBOL Estándar donde pgm2 o pgm3 en el ejemplo anterior se llama utilizando CALL *identificador* de otro programa.
- Si NOSEPOBJ está en vigor, a los archivos de objeto se les asigna el nombre del archivo de origen pero con el sufijo .o. Si SEPOBJ está en vigor, los nombres de los archivos de objeto se basan en el nombre PROGRAM-ID con sufijo .o.
- Los programas llamados utilizando CALL *identificador* deben estar referenciados por los nombres de los archivos de objeto (en lugar de los nombres PROGRAM-ID) donde PROGRAM-ID y el nombre de archivo de objeto no coinciden.

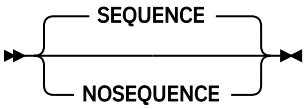
Debe asignar al archivo objeto un nombre de archivo válido para la plataforma y el sistema de archivos.

SECUENCIA

Cuando se utiliza SEQUENCE, el compilador examina las columnas 1 a 6 para comprobar que las sentencias de origen se ordenan en orden ascendente según su secuencia de clasificación ASCII . El compilador emite un mensaje de diagnóstico si alguna sentencia no está en orden ascendente.

Las sentencias fuente con espacios en blanco en las columnas 1 a 6 no participan en esta comprobación de secuencia y no dan como resultado mensajes.

Sintaxis de la opción SEQUENCE



El valor predeterminado es: SEQUENCE

Las abreviaturas son: SEQ | NOSEQ

Si utiliza sentencias COPY con la opción SEQUENCE en vigor, asegúrese de que los campos de secuencia del programa fuente y los campos de secuencia del libro de copias estén coordinados.

Si utiliza NUMBER y SEQUENCE, la secuencia se comprueba de acuerdo con el orden de clasificación numérico, en lugar de ASCII.

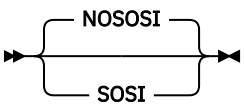
Utilice NOSEQUENCE para suprimir esta comprobación y los mensajes de diagnóstico.

Tareas relacionadas
“Búsqueda de problemas de secuencia de línea” en la página 324

SOSI

La opción SOSI afecta al tratamiento de los valores X'1E' y X'1F' en comentarios; literales alfanuméricos, nacionales y DBCS ; y en palabras DBCS definidas por el usuario.

Sintaxis de la opción SOSI



El valor predeterminado es: NOSOSI

Las abreviaturas son: Ninguna

NOSOSI
Con NOSOSI, las posiciones de caracteres que tienen valores X'1E' y X'1F' se tratan como caracteres de datos.

NOSOSI se ajusta a 85 COBOL Estándar.

SOSI
Con SOSI, los caracteres de control de desplazamiento a teclado ideográfico (SO) y de desplazamiento a teclado estándar (SI) delimitan series de caracteres DBCS ASCII en programas fuente COBOL. Los caracteres SO y SI tienen los valores codificados de X'1E' y X'1F', respectivamente.

Los caracteres SO y SI no tienen ningún efecto en el código fuente de COBOL para Linux , excepto para actuar como marcadores de posición para los caracteres SO y SI de DBCS de host para garantizar un manejo de datos adecuado cuando los archivos remotos se convierten de EBCDIC a ASCII.

Cuando la opción SOSI está en vigor, además de las reglas existentes para COBOL para Linux, se aplican las reglas siguientes:

- Todas las series de caracteres DBCS (en palabras definidas por el usuario, literales DBCS, literales alfanuméricos, literales nacionales y en comentarios) deben estar delimitadas por los caracteres SO y SI.
- Las palabras definidas por el usuario no pueden contener caracteres DBCS y SBCS a la vez.
- La longitud máxima de una palabra DBCS definida por el usuario es de 14 caracteres DBCS.
- Las letras alfabéticas en mayúscula de doble byte no son equivalentes a las letras en minúscula de doble byte correspondientes cuando se utilizan en palabras definidas por el usuario.
- Una palabra DBCS definida por el usuario debe contener al menos una letra que no tenga su equivalente en una representación de un solo byte.
- Las representaciones de doble byte de caracteres de un solo byte para A-Z, a-z, 0-9, el guión (-)y el subrayado (_) pueden incluirse dentro de una palabra DBCS definida por el usuario. Las reglas aplicables a estos caracteres en representación de un solo byte se aplican a estos caracteres en representación de doble byte. Por ejemplo, en una palabra definida por el usuario, el guión no puede aparecer como el primer o el último carácter, y el subrayado no puede aparecer como el primer carácter.
- Para los literales DBCS y nacionales que contienen valores X'1E' o X'1F' , se aplican las reglas siguientes cuando la opción de compilador SOSI está en vigor:
  - Las posiciones de caracteres con X'1E' y X'1F' se tratan como caracteres SO y SI.
  - Las posiciones de caracteres con X'1E' y X'1F' se incluyen en la serie de caracteres en notación hexadecimal nacional y se eliminan en notación básica.
- Para literales alfanuméricos que contienen valores X'1E' o X'1F' , se aplican las reglas siguientes cuando la opción de compilador SOSI está en vigor:
  - Las posiciones de caracteres con X'1E' y X'1F' se tratan como caracteres SO y SI.
  - Las posiciones de caracteres con X'1E' y X'1F' se incluyen en la serie de caracteres en notación hexadecimal y se eliminan en notación básica y terminada en nulo.
- Para incluir comillas DBCS dentro de un literal Ndelimitado por comillas, utilice dos comillas DBCS consecutivas para representar una sola comilla DBCS. No incluya una sola comilla DBCS en un literal Nsi el literal está delimitado por comillas. La misma regla se aplica a las comillas simples.
- Los registros especiales SHIFT-OUT y SHIFT-IN se definen con X'0E' y X'0F' independientemente de si la opción SOSI está en vigor.

En general, los programas COBOL de host que son sensibles a los valores codificados para los caracteres SO y SI no tendrán el mismo comportamiento en la estación de trabajo de Linux .

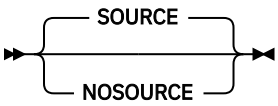
Tareas relacionadas
"Manejo de diferencias en series ASCII multibyte y EBCDIC DBCS" en la *Guía de migración*

Referencias relacionadas
Cadenas de caracteres (*COBOL for Linux en x86 Consulta de lenguaje*)

fuente

Utilice SOURCE para obtener una lista del programa fuente. Este listado incluirá las sentencias incorporadas por las sentencias PROCESS o COPY .

Sintaxis de la opción SOURCE



El valor predeterminado es: SOURCE

Las abreviaturas son: S | NOS

Debe especificar SOURCE si desea mensajes incorporados en el listado fuente.

Utilice NOSOURCE para suprimir el código fuente del listado de salida del compilador.

Si desea limitar la salida de SOURCE , utilice sentencias *CONTROL SOURCE o NOSOURCE en PROCEDURE DIVISION. Las sentencias de origen que siguen a una sentencia *CONTROL NOSOURCE no se incluyen en el listado hasta que una sentencia *CONTROL SOURCE posterior vuelve a conmutar la salida al formato SOURCE normal.

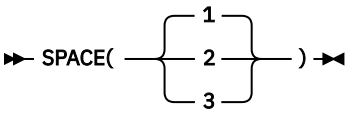
"Ejemplo: salida MAP" en la página 387

Referencias relacionadas
*CONTROL (*CBL) (*COBOL for Linux en x86 Consulta de lenguaje*)

ESPACIO

Utilice SPACE para seleccionar un espacio único, doble o triple en el listado de código fuente.

Sintaxis de la opción SPACE



El valor predeterminado es: SPACE(1)

Las abreviaturas son: Ninguna

SPACE sólo tiene significado cuando la opción de compilador SOURCE está en vigor.

Referencias relacionadas
"fuente" en la página 297

SPILL

Esta opción especifica el número de KB reservados para el área de vertido de registro. Si el programa que se está compilando es muy complejo o grande, esta opción puede ser necesaria.

Sintaxis de la opción SPPILL

El valor predeterminado es: SPILL(512)

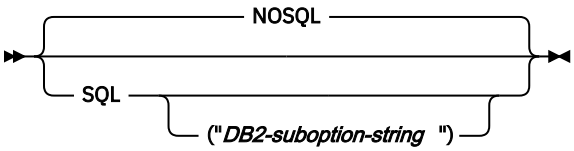
Las abreviaturas son: Ninguna

El tamaño del vertido, *n*, es cualquier entero entre 96 y 32704.

SQL

Utilice la opción de compilador SQL para habilitar el coprocesador de Db2 y para especificar las subopciones de Db2 . Debe especificar la opción SQL si el programa fuente COBOL contiene sentencias SQL y el programa no ha sido procesado por el precompilador Db2 .

Sintaxis de la opción SQL



El valor predeterminado es: NOSQL

Las abreviaturas son: Ninguna

Si NOSQL está en vigor, las sentencias SQL que se encuentren en el programa fuente se diagnosticarán y se descartarán.

Utilice comillas o comillas simples para delimitar la serie de subopciones de Db2 .

Puede utilizar la sintaxis mostrada anteriormente en la sentencia CBL o PROCESS . Si utiliza la opción SQL en el mandato cob2, sólo se puede utilizar la comilla simple (') como delimitador de serie de subopción: `-q"SQL('suboptions')"`.

Nota: El compilador interpreta determinados caracteres de script de shell como se indica a continuación:

- Un signo igual (=) se interpreta como un paréntesis izquierdo, (
- Un signo de dos puntos (:) se interpreta como un paréntesis derecho,)
- Un subrayado (_) se interpreta como una comilla simple (')

Puede añadir un carácter de escape de barra inclinada invertida (\) para evitar la interpretación y, por lo tanto, para pasar caracteres en las series. Si desea que la barra inclinada invertida (\) se represente a sí misma (en lugar de como un carácter de escape), utilice la barra inclinada invertida doble (\\).

Por ejemplo, si desea trabajar con el coprocesador integrado de Db2 y utilizar la opción de precompilación DEFERRED_PREPARE , especifique la opción SQL como se indica a continuación:

```
SQL('... DEFERRED\_PREPARE ...')
```

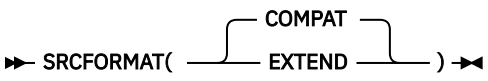
Tareas relacionadas
Capítulo 17, “Programación para un entorno Db2”, en la página 399
“Compilación con la opción SQL” en la página 403
“Separación de subopciones de Db2” en la página 404

Referencias relacionadas
“Opciones de compilador en conflicto” en la página 267

FORMATOORIGEN

Utilice SRCFORMAT para indicar si el origen COBOL se ajusta al formato de origen fijo de 72 columnas o al formato de origen ampliado de 252 columnas.

Sintaxis de la opción SRCFORMAT



El valor predeterminado es: SRCFORMAT(COMPAT)

Las abreviaturas son: SF(C|E)

SRCFORMAT(COMPAT) indica que cada línea fuente de la entrada de compilación primaria y de cualquier texto COPY incluido termina en la columna 72. Si una línea de origen es más corta que 72 bytes, los caracteres de espacio se añaden lógicamente a la línea de origen hasta un máximo de 72 bytes. Si una línea fuente tiene más de 72 bytes, sólo se utilizan los primeros 72 bytes como fuente de programa. (Se supone que los bytes 73 a 80, si se proporcionan, contienen números de serie; se imprimen en el listado del compilador, pero por lo demás se ignoran.)

SRCFORMAT(EXTEND) indica que cada línea fuente de la entrada de compilación primaria y de cualquier texto COPY incluido finaliza en la columna 252. Si una línea de origen es inferior a 252 bytes, los caracteres de espacio se añaden lógicamente a la línea de origen hasta un máximo de 252 bytes. Si una línea fuente tiene más de 252 bytes, sólo se utilizan los primeros 252 bytes como fuente de programa; el resto se ignora. (En el formato de origen ampliado, no hay ninguna disposición para los números de serie.)

En cualquiera de los formatos, las columnas 1 a 6 se interpretan como números de secuencia.

Especificación de opción: No puede especificar la opción SRCFORMAT en una sentencia PROCESS (o CBL). Sólo puede especificarlo de una de las maneras siguientes:

- Como opción en el mandato cob2
- En la variable de entorno COBOPT
- En el atributo compopts del archivo de configuración (.cfg)

Un programa de utilidad de conversión de origen, scu, está disponible para ayudar a convertir código fuente COBOL que no sea deIBM o de formato libre para que COBOL para Linuxpueda compilarlo. Para ver un resumen de las funciones de scu , escriba el mandato scu -h. Para obtener más detalles, consulte la página man para scu, o consulte la referencia relacionada adecuada.

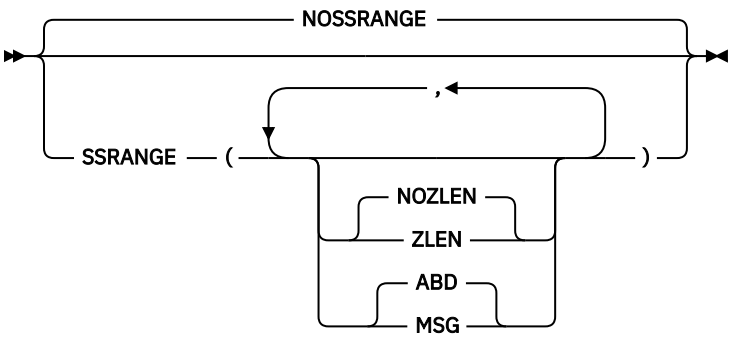
Restricción: El formato de origen ampliado no es compatible con el precompilador Db2 autónomo o el conversor CICS independiente.

Referencias relacionadas
"Stanzas en el archivo de configuración" en la página 244
"Opciones cob2" en la página 248
Formato de referencia (*COBOL for Linux en x86 Consulta de lenguaje*)
Programa de utilidad de conversión de origen (scu) (*COBOL for Linux en x86 Consulta de lenguaje*)

SRANGE

Utilice SSRANGE para generar código que compruebe las referencias de almacenamiento fuera de rango.

Sintaxis de la opción SSRANGE



El valor predeterminado es: NOSSRANGE

El valor predeterminado de la subopción es: NOZLEN,ABD si sólo se especifica SSRANGE .

Las abreviaturas son: SSR | NOSSR

SSRANGE genera código que comprueba si los subíndices, incluidos los subíndices de ALL , intentan hacer referencia a áreas fuera de la región de sus tablas asociadas. No se comprueba individualmente la validez de cada subíndice o índice. En su lugar, se comprueba la dirección efectiva para asegurarse de que no hace referencia fuera de la tabla.

Si especifica SSRANGE sin subopciones, se aceptará como una especificación de SSRANGE(NOZLEN,ABD).

Nota: Si la opción SSRANGE está en vigor, el compilador generará comprobaciones de rango y las comprobaciones siempre se realizarán en tiempo de ejecución. No puede inhabilitar las comprobaciones de rango compilado en tiempo de ejecución especificando la opción de tiempo de ejecución CHECK(OFF).

Los elementos de longitud variable también se comprueban para asegurarse de que las referencias están dentro de su longitud máxima definida.

Las expresiones de modificación de referencia se comprueban para asegurarse de que:

- La posición inicial es mayor o igual que 1.
- La posición inicial no es mayor que la longitud actual del elemento de datos de asunto.
- La posición inicial y el valor de longitud (si se especifica) no hacen referencia a un área más allá del final del elemento de datos de asunto.
- El valor de longitud (si se especifica) es mayor o igual que 1.

Las subopciones ZLEN y NOZLEN controlan cómo el compilador comprueba las longitudes de modificación de referencia:

- Si ZLEN está en vigor, el compilador generará código para asegurarse de que las longitudes de modificación de referencia sean mayores o iguales a cero. Las especificaciones de modificación de referencia de longitud cero no obtendrán un error SSRANGE en tiempo de ejecución.
- Si NOZLEN está en vigor, el compilador generará código para asegurarse de que las longitudes de modificación de referencia sean mayores o iguales que 1. Las especificaciones de modificación de referencia de longitud cero obtendrán un error SSRANGE en tiempo de ejecución. Esto es compatible con el comportamiento de SSRANGE en versiones anteriores de COBOL.

Las subopciones MSG y ABD controlan el comportamiento en tiempo de ejecución del programa COBOL cuando falla una comprobación de rango.

- Si MSG está en vigor y falla una comprobación de rango, se emitirá un mensaje de aviso de tiempo de ejecución. Esto significa que el programa continuará ejecutándose y podría potencialmente identificar otras condiciones fuera de rango.

- Si ABD está en vigor y falla una comprobación de rango, la primera condición fuera de rango dará como resultado un mensaje de error de tiempo de ejecución y el programa terminará anormalmente. Puede encontrar la siguiente condición potencial fuera de rango arreglando la primera condición fuera de rango y, a continuación, recompilando y ejecutando de nuevo el programa. Para identificar todas las demás condiciones potenciales fuera de rango, es posible que tenga que repetir este proceso varias veces.

Para grupos sin límites o sus elementos subordinados, la comprobación sólo se realiza para expresiones de modificación de referencia. Las referencias con subíndice o indexadas a tablas subordinadas a un grupo ilimitado no se comprueban.

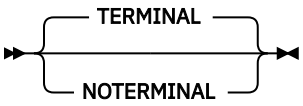
Conceptos relacionados
"Modificadores de referencia" en la página 103

Tareas relacionadas
"Comprobación de rangos válidos" en la página 324

TERMINAL

Utilice TERMINAL para enviar mensajes de progreso y diagnóstico aldispositivo de visualización .

Sintaxis de la opción TERMINAL



El valor predeterminado es: TERMINAL

Las abreviaturas son: TERM | NOTERM

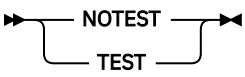
Utilice NOTERMINAL si no desea esta salida adicional.

TEST

Utilice TEST para generar código de objeto que contenga información de símbolos y sentencias que permita al depurador realizar una depuración simbólica a nivel de fuente.

También se crea un archivo MDECK (.dek) que contiene una copia del origen de entrada actualizado después del proceso de biblioteca (es decir, el resultado de las sentencias COPY, BASIS, REPLACEy EXEC SQL INCLUDE) para que lo utilice el depurador.

Sintaxis de la opción TEST



El valor predeterminado es: NOTEST

Las abreviaturas son: Ninguna

Utilice NOTEST si no desea generar código de objeto que tenga información de depuración. Los programas compilados con NOTEST se ejecutan con el depurador, pero tienen un soporte de depuración limitado.

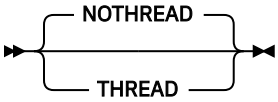
Si utiliza la cláusula WITH DEBUGGING MODE , la opción TEST está desactivada. TEST aparecerá en la lista de opciones, pero se emite un mensaje de diagnóstico para avisarle de que debido al conflicto, TEST no estaba en vigor.

Tareas relacionadas
"Depuración utilizando IBM Debug for Linux en x86" en la página 328

HEBRA

La opción THREAD se acepta y se ignora. Ya no es necesario indicar que se va a habilitar un programa COBOL para su ejecución en una unidad de ejecución que tiene varias hebras.

Sintaxis de la opción THREAD



El valor predeterminado es: NOTHREAD

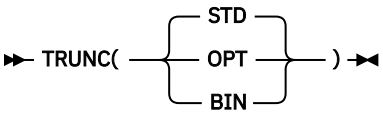
Las abreviaturas son: Ninguna

Tareas relacionadas
"Compilación desde la línea de mandatos" en la página 240

TRUNC

TRUNC afecta a la forma en que se truncan los datos binarios durante los movimientos y las operaciones aritméticas.

Sintaxis de la opción TRUNC



El valor predeterminado es: TRUNC(STD)

Las abreviaturas son: Ninguna

TRUNC no tiene ningún efecto en los elementos de datos COMP-5 ; los elementos COMP-5 se manejan como si TRUNC(BIN) estuviera en vigor independientemente de la subopción TRUNC especificada.

TRUNC(STD)
TRUNC(STD) sólo se aplica a los campos de recepción de USAGE BINARY en sentencias MOVE y expresiones aritméticas. Cuando TRUNC(STD) está en vigor, el resultado final de una expresión aritmética, o el campo de envío de la sentencia MOVE , se trunca en el número de dígitos de la cláusula PICTURE del campo de recepción BINARY .

TRUNC(OPT)
TRUNC(OPT) es una opción de rendimiento. Cuando TRUNC(OPT) está en vigor, el compilador presupone que los datos se ajustan a las especificaciones PICTURE en los campos de recepción de USAGE BINARY en sentencias MOVE y expresiones aritméticas. Los resultados se manipulan de la forma más óptima, ya sea truncando al número de dígitos de la cláusula PICTURE o al tamaño del campo binario en el almacenamiento (media palabra, palabra completa o palabra doble).

Sugerencia: Utilice la opción TRUNC(OPT) sólo si está seguro de que los datos que se están moviendo a las áreas binarias no tendrán un valor con una precisión mayor que la definida por la cláusula PICTURE para el elemento binario. De lo contrario, podrían producirse resultados imprevisibles. Este truncamiento se realiza de la manera más eficiente posible; por lo tanto, los resultados dependen de la secuencia de código particular generada. No es posible predecir el truncamiento sin ver la secuencia de código generada para una sentencia determinada.

TRUNC(BIN)

La opción TRUNC(BIN) se aplica a todos los lenguajes COBOL que procesan datos USAGE BINARY . Cuando TRUNC(BIN) está en vigor, todos los elementos binarios (USAGE COMP, COMP-4o BINARY) se manejan como elementos binarios de hardware nativos, es decir, como si cada uno de ellos se declarara individualmente USAGE COMP-5:

- Los campos de recepción de BINARY sólo se truncan en los límites de media palabra, palabra completa o palabra doble.
- Los campos de envío de BINARY se manejan como medias palabras, palabras completas o palabras dobles cuando el receptor es numérico; TRUNC(BIN) no tiene ningún efecto cuando el receptor no es numérico.
- El contenido binario completo de los campos es significativo.
- DISPLAY convertirá todo el contenido de los campos binarios sin truncamiento.

Recomendaciones: TRUNC(BIN) es la opción recomendada para programas que utilizan valores binarios establecidos por otros productos. Otros productos, como por ejemplo Db2 y C/C++, pueden colocar valores en elementos de datos binarios COBOL que no se ajusten a la cláusula PICTURE de los elementos de datos. Puede utilizar TRUNC(OPT) con programas CICS siempre que los datos se ajustan a la cláusula PICTURE para los elementos de datos de BINARY .

USAGE COMP-5 tiene el efecto de aplicar el comportamiento de TRUNC(BIN) a elementos de datos individuales. Por lo tanto, puede evitar la sobrecarga de rendimiento de utilizar TRUNC(BIN) para cada elemento de datos binarios especificando COMP-5 sólo en algunos de los elementos de datos binarios, como los elementos de datos que se pasan a programas no COBOL u otros productos y subsistemas. El uso de COMP-5 no se ve afectado por la subopción TRUNC en vigor.

Literales grandes en cláusulas VALUE : cuando se utiliza la opción de compilador TRUNC(BIN), literales numéricos especificados en cláusulas VALUE para elementos de datos binarios (COMP, COMP-4, o BINARY) generalmente puede contener un valor de magnitud hasta la capacidad de la representación binaria nativa (2, 4 u 8 bytes) en lugar de limitarse al valor implícito por el número de 9s en la cláusula PICTURE .

Nota: Cuando TRUNC(BIN) y NUMCHECK(BIN) están en vigor y se genera un mensaje de error o una terminación anómala, si tiene previsto cambiar a TRUNC(STD|OPT) más adelante para obtener un mejor rendimiento, debe corregir los datos; si no es así, puede desactivar NUMCHECK(BIN) para reducir el tiempo de ejecución de la aplicación y evitar un mensaje de error o una terminación anómala.

Ejemplo 1 de TRUNC

```
01  BIN-VAR     PIC S99 USAGE BINARY.
. . .
    MOVE 123451 to BIN-VAR
```

La tabla siguiente muestra los valores de los elementos de datos después de la sentencia MOVE .

Elemento de datos	Decimal	Hexadecimal	Mostrar
Remitente	123451	3B | E2 | 01 | 00	123451
Receptor TRUNC(STD)	51	33 | 00	51
Receptor TRUNC(OPT)	-7621	3B | E2	2J
Receptor TRUNC(BIN)	-7621	3B | E2	762J

Se asigna una media palabra de almacenamiento para BIN-VAR. El resultado de esta sentencia MOVE si el programa se compila con la opción TRUNC(STD) es 51; el campo se trunca para ajustarse a la cláusula PICTURE .

Si compila el programa con TRUNC(BIN), el resultado de la sentencia MOVE es -7621. La razón del resultado inusual es que se truncan los dígitos de orden superior distintos de cero. Aquí, la secuencia de código generada simplemente movería la cantidad de media palabra inferior X'E23B' al receptor. Debido a que el nuevo valor truncado se desborda en el bit de signo de la media palabra binaria, el valor se convierte en un número negativo.

Es mejor no compilar esta sentencia MOVE con TRUNC(OPT), porque 123451 tiene una precisión mayor que la cláusula PICTURE para BIN-VAR. Con TRUNC(OPT), los resultados vuelven a ser -7621. Esto se debe a que el mejor rendimiento se ha obtenido al no realizar un truncamiento decimal.

Ejemplo 2 de TRUNC

```
01  BIN-VAR     PIC 9(6)  USAGE BINARY
. . .
    MOVE 1234567891 to BIN-VAR
```

La tabla siguiente muestra los valores de los elementos de datos después de la sentencia MOVE .

Elemento de datos	Decimal	Hexadecimal	Mostrar
Remitente	1234567891	D3 | 02 | 96 | 49	1234567891
Receptor TRUNC(STD)	567891	53 | AA | 08 | 00	567891
Receptor TRUNC(OPT)	567891	53|AA|08|00	567891
Receptor TRUNC(BIN)	1234567891	D3 | 02 | 96 | 49	1234567891

Cuando se especifica TRUNC(STD), los datos de envío se truncan en seis dígitos enteros para ajustarse a la cláusula PICTURE del receptor BINARY .

Cuando especifica TRUNC(OPT), el compilador presupone que los datos de envío no son mayores que la precisión de la cláusula PICTURE del receptor BINARY . La secuencia de código más eficiente en este caso es el truncamiento como si TRUNC(STD) estuviera en vigor.

Cuando especifica TRUNC(BIN), no se produce ningún truncamiento porque todos los datos de envío se ajustan a la palabra completa binaria asignada para BIN-VAR.

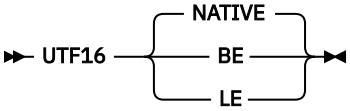
Conceptos relacionados
"Formatos para datos numéricos" en la página 39

Referencias relacionadas
cláusula VALUE (*COBOL for Linux en x86 Consulta de lenguaje*)

UTF16

UTF16 especifica el formato de representación de los elementos de datos UTF-16 .

Sintaxis de la opción UTF16



El valor predeterminado es: UTF16(NATIVE)

Las abreviaturas son: Ninguna

Especifique UTF16(NATIVE) para utilizar el formato de representación nativo UTF-16 de la plataforma. Para COBOL para Linux, este es el formato *little-endian* (dígito menos significativo en la dirección más baja).

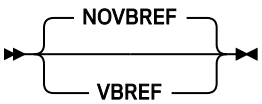
UTF16(BE) indica que los elementos de datos UTF-16 se representan de forma coherente con IBM Z, es decir, en formato *big-endian* (dígito más significativo en la dirección más baja).

UTF16(LE) indica que los elementos de datos UTF-16 se representan en formato *little-endian* (dígito menos significativo en la dirección más baja).

VBREF

Utilice VBREF para obtener una referencia cruzada entre todas las sentencias utilizadas en el programa de origen y los números de línea en los que se utilizan. VBREF también genera un resumen del número de veces que se ha utilizado cada sentencia en el programa.

Sintaxis de la opción VBREF



El valor predeterminado es: NOVBREF

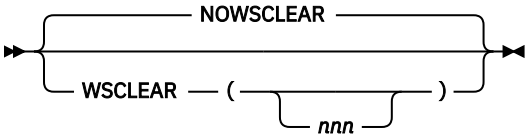
Las abreviaturas son: Ninguna

Utilice NOVBREF para una compilación más eficiente.

WCLEAR

Utilice WSCLEAR para borrar los elementos de datos noEXTERNAL de un programa en WORKING-STORAGE a ceros binarios durante la inicialización. El almacenamiento se borra antes de que se apliquen las cláusulas VALUE .

Sintaxis de la opción WSCLEAR



El valor predeterminado es: NOWSCLEAR

Las abreviaturas son: Ninguna

Utilice NOWSCLEAR para omitir el proceso de borrado de almacenamiento.

nnn es cualquier entero de 0 a 255. WSCLEAR sin la subopción es el mismo que WSCLEAR(*0*).

Puede especificar la opción WSCLEAR en una línea de mandatos o en una sentencia COBOL. Sin embargo, la opción WSCLEAR que se especifica en la sentencia COBOL tiene prioridad sobre la opción especificada en la línea de mandatos.

- Para especificar WSCLEAR con la subopción *nnn* en la línea de mandatos, utilice el formato de -qwsclear*(nnn)*. En este caso, el procesador de mandatos explora los caracteres que están en el rango de 0 a 255 únicamente, y se ignoran otros caracteres excesivos. No se emite ningún mensaje de error.

  Por ejemplo, si especifica -qwsclear*(99999)*, el procesador de mandatos sólo toma WSCLEAR*(99)* .
- Para especificar WSCLEAR con la subopción *nnn* en sentencias COBOL, utilice el formato WSCLEAR*(nnn)* .

Si especifica WSCLEAR(*nnn*), el valor de byte representado por *nnn* se utiliza para inicializar cada byte de datos WORKING-STORAGE en un valor específico. Esto sólo se aplica a los elementos de datos que no tienen especificado un atributo VALUE.

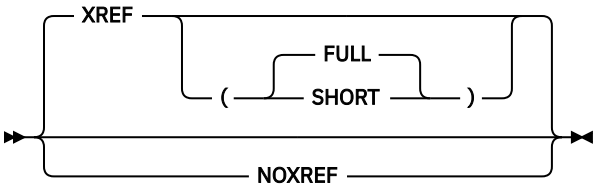
Consideraciones sobre el rendimiento: Si utiliza WSCLEAR y le preocupa el tamaño o el rendimiento de un programa objeto, utilice también OPTIMIZE(FULL). Al hacerlo, se indica al compilador que elimine todos los elementos de datos no referenciados del DATA DIVISION, lo que acelerará la inicialización.

Referencias relacionadas
"OPTIMIZAR" en la página 291

XREF

Utilice XREF para generar un listado de referencias cruzadas ordenado.

Sintaxis de la opción XREF



El valor predeterminado es: XREF(FULL)

Las abreviaturas son: X | NOX

Puede elegir XREF, XREF(FULL) o XREF(SHORT). Si especifica XREF sin ninguna subopción, XREF(FULL) estará en vigor.

Una sección del listado muestra todos los nombres de programa, nombres de datos y nombres de procedimiento a los que se hace referencia en el programa, y los números de línea donde se definen dichos nombres. Se identifican los nombres de programa externos.

"Ejemplo: Salida XREF: referencias cruzadas de nombre de datos" en la página 391
"Ejemplo: Salida XREF: referencias cruzadas de nombre de programa" en la página 392

También se incluye una sección que hace referencia cruzada a sentencias COPY o BASIS en el programa con los conjuntos de datos de los que se han obtenido los libros de copias asociados.

"Ejemplo: Salida XREF: COPY/BASE referencias cruzadas" en la página 392

Los nombres se listan en el orden del orden de clasificación indicado por el valor de entorno local. Este orden se utiliza si los nombres están en caracteres de un solo byte o contienen caracteres de varios bytes (como DBCS).

Si utiliza XREF y SOURCE, la información de referencia cruzada de nombre de datos y nombre de procedimiento se imprime en la misma línea que el origen original. Las referencias de número de línea u otra información aparecen en la parte derecha de la página de listado. A la derecha de las líneas fuente que hacen referencia a una función intrínseca, las letras IFN se imprimen con el número de línea de las ubicaciones donde se definen los argumentos de la función. La información incluida en las referencias incorporadas le permite saber si un identificador no está definido (UND) o está definido más de una vez (DUP), si los elementos están definidos implícitamente (IMP) (como registros especiales o constantes figurativas) o si un nombre de programa es externo (EXT).

Si utiliza XREF y NOSOURCE, sólo obtendrá el listado de referencias cruzadas ordenado.

XREF(SHORT) sólo imprime los elementos de datos referenciados explícitamente en el listado de referencias cruzadas. XREF(SHORT) se aplica a nombres de datos y nombres de procedimiento multibyte , así como a nombres de un solo byte.

NOXREF suprime este listado.

Notas de uso

- Los nombres de grupo utilizados en una sentencia MOVE CORRESPONDING se encuentran en el listado XREF . También se listan los nombres elementales de esos grupos.
- En el listado de nombres de datos XREF , los números de línea precedidos por la letra M indican que el elemento de datos se modifica explícitamente mediante una sentencia en esa línea.
- Los listados de XREF toman almacenamiento adicional.

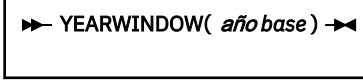
Conceptos relacionados
Capítulo 16, “depuración”, en la página 319

Tareas relacionadas
“Obtención de listados” en la página 382

VENTANA

Utilice YEARWINDOW para especificar el primer año de la ventana de 100 años (la *ventana de siglo*) que el compilador COBOL debe aplicar al proceso de campos de fecha con ventanas.

Sintaxis de la opción YEARWINDOW



El valor predeterminado es: YEARWINDOW(1900)

Las abreviaturas son: YW

año base representa el primer año de la ventana de 100 años. Debe especificarlo con uno de los valores siguientes:

- Un entero decimal sin signo entre 1900 y 1999.

  Un entero sin signo especifica el año de inicio de una ventana fija. Por ejemplo, YEARWINDOW(1930) indica la ventana de siglo 1930-2029.
- Un entero negativo de -1 a -99.

  Un entero negativo indica una ventana deslizante. El primer año de la ventana se calcula añadiendo el entero negativo al año actual. Por ejemplo, YEARWINDOW(-80) indica que el primer año de la ventana de siglo es 80 años antes del año en el que se ejecuta el programa.

Notas de uso

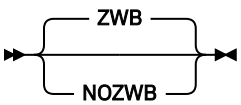
- La opción YEARWINDOW no tiene ningún efecto a menos que la opción DATEPROC también esté en vigor.
- En tiempo de ejecución, deben cumplirse dos condiciones:
  - La ventana de siglo debe tener su año de inicio en 1900s.
  - El año en curso debe estar dentro de la ventana del siglo para la unidad de compilación.

  Por ejemplo, si el año actual es 2010, la opción DATEPROC está en vigor y utiliza la opción YEARWINDOW(1900) , el programa terminará con un mensaje de error.

ZWB

Si compila utilizando ZWB, el compilador elimina el signo de un campo decimal con zona (DISPLAY) con signo antes de comparar este campo con un campo elemental alfanumérico durante la ejecución.

Sintaxis de la opción ZWB



El valor predeterminado es: ZWB

Las abreviaturas son: Ninguna

Si el elemento decimal con zona es un elemento escalado (es decir, contiene el símbolo P en su serie PICTURE), las comparaciones que utilizan el elemento decimal no se verán afectadas por ZWB. A estos elementos siempre se les elimina el signo antes de realizar la comparación con un campo alfanumérico.

ZWB afecta a la forma en que se ejecuta un programa. El mismo programa COBOL puede producir resultados diferentes en función del valor de esta opción.

Utilice NOZWB si desea probar los campos numéricos de entrada para SPACES.

Capítulo 14. Sentencias de direccionamiento de compilador

Varias sentencias de direccionamiento de compilador le ayudan a dirigir la compilación del programa.

Estas son las sentencias de dirección del compilador:

BASIS sentencia
Esta sentencia de biblioteca de programa fuente ampliada proporciona un programa COBOL completo como fuente para una compilación. Para reglas de formación y proceso, consulte la descripción de *text-name* para la sentencia COPY .

***CONTROL (*CBL) sentencia**
Esta sentencia de dirección de compilador suprime de forma selectiva o permite que se genere la salida . Las palabras clave *CONTROL y *CBL son sinónimas.

DirectivaCALLINTERFACE
Esta directiva de compilador especifica el convenio de interfaz para llamadas e indica si se van a generar descriptores de argumento. El convenio especificado con >>CALLINTERFACE está en vigor hasta que se realice otra especificación >>CALLINTERFACE . >>CALLINT es una abreviatura de >>CALLINTERFACE.

>>CALLINTERFACE sólo se puede utilizar en PROCEDURE DIVISION.

La sintaxis y el uso de la directiva >>CALLINTERFACE son similares a los de la opción de compilador CALLINT . Las excepciones son:

- La sintaxis de la directiva no incluye paréntesis.
- La directiva se puede aplicar a las llamadas seleccionadas tal como se describe a continuación.
- La sintaxis de la directiva incluye la palabra clave DESCRIPTOR y sus variantes.

Si especifica >>CALLINT sin subopciones, el convenio de llamada utilizado lo determina la opción de compilador CALLINT .

Sólo**DESCRIPTOR :** la directiva >>CALLINT se trata como un comentario excepto para estos formatos:

- >>CALLINT SYSTEM DESCRIPTOR, o de forma equivalente >>CALLINT DESCRIPTOR
- >>CALLINT SYSTEM NODESCRIPTOR, o de forma equivalente >>CALLINT NODESCRIPTOR

Estas directivas activan o desactivan DESCRIPTOR ; se ignora SYSTEM .

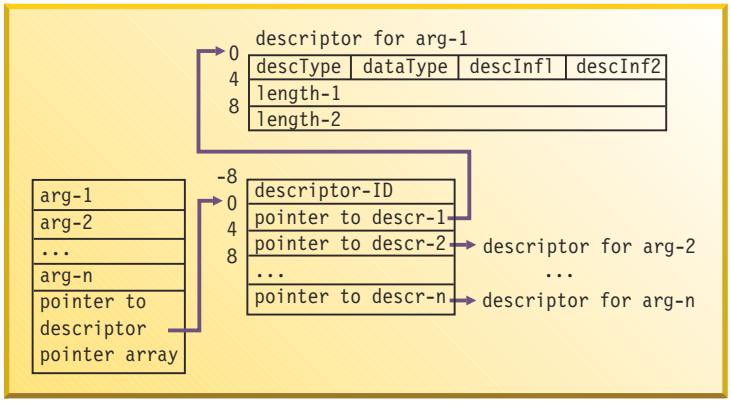
La directiva >>CALLINT se puede especificar en cualquier lugar en el que se pueda especificar una sentencia de procedimiento COBOL. Por ejemplo, esta sintaxis es válida:

```
MOVE 3 TO
>>CALLINTERFACE SYSTEM
RETURN-CODE.
```

El efecto de >>CALLINT está limitado al programa actual. Un programa anidado o un programa compilado en el mismo lote hereda el convenio de llamada especificado en la opción de compilador CALLINT , pero no un convenio especificado por la directiva de compilador >>CALLINT .

Si está escribiendo una rutina que se va a llamar con >>CALLINT SYSTEM DESCRIPTOR, este es el mecanismo de paso de argumentos:

CALL "PROGRAM1" USING arg-1, arg-2, ... arg-n



puntero a descr-n
Apunta al descriptor para el argumento específico; 0 si no existe ningún descriptor para el argumento.

ID-descriptor
Establézcalo en COBDESC0 para identificar esta versión del descriptor, lo que permite un posible cambio en el formato de entrada del descriptor en el futuro.

tipoDesc
Establézcalo en X'02 ' (descElmt) para un elemento de datos elemental de USAGE DISPLAY con PICTURE X(*n*) o USAGE DISPLAY-1 con PICTURE G(*n*) o N(*n*). Para todos los demás (campos numéricos, estructuras, tablas), establézcalo en X'00 '.

tipoDatos
Establezca lo siguiente:

- descType = X'00 ': tipoDatos = X'00'
- descType = X'02 'y USAGE es DISPLAY: dataType = X'02' (typeChar)
- descType = X'02 'y USAGE es DISPLAY-1: dataType = X'09' (typeGChar)

descInf1
Establézcalo siempre en X'00 '.

descInf2
Establezca lo siguiente:

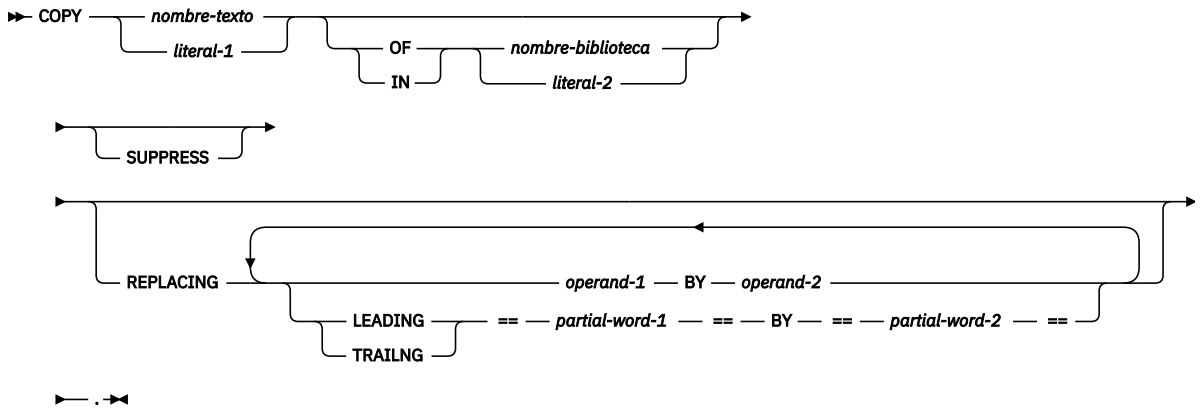
- Si descType = X'00 '; descInf2 = X'00'
- Si descType = X'02 ':
  - Si la opción CHAR(EBCDIC) está en vigor y el argumento no está definido con la opción NATIVE en la cláusula USAGE : descInf2 = X'40 '
  - De lo contrario: descInf2 = X'00 '

length-1
En el descriptor de argumento es la longitud del argumento para un argumento de longitud fija o la longitud actual para un elemento de longitud variable.

length-2
La longitud máxima del argumento si el argumento es un elemento de longitud variable. Para un argumento de longitud fija, length-2 es igual a length-1.

COPY sentencia

Sintaxis de la sentencia COPY



Esta sentencia de direccionamiento de compilador coloca texto preescrito en un programa COBOL.

No es necesario que *text-name* ni *library-name* sean exclusivos dentro de un programa. Pueden ser idénticas a otras palabras definidas por el usuario en el programa.

Debe especificar un *nombre-texto* (el nombre de un libro de copias) que contenga el texto preescrito; por ejemplo, `COPY my-text`. Puede calificar *text-name* con un *library-name*; por ejemplo, `COPY my-text of inventory-lib`. Si *nombre-texto* no está calificado, se presupone un *nombre-biblioteca* de SYSLIB .

nombre-biblioteca

Si especifica *nombre-biblioteca* como un literal, el contenido del literal se trata como la vía de acceso real. Si especifica *nombre-biblioteca* como una palabra definida por el usuario, el nombre se utiliza como una variable de entorno y el valor de la variable de entorno se utiliza para la vía de acceso para localizar el libro de copias. Para especificar varios nombres de vía de acceso, delícelos con un signo de dos puntos (:).

Si no especifica *nombre-biblioteca*, la vía de acceso utilizada es la que se describe en *nombre-texto*.

nombre-texto

Si especifica *nombre-texto* como un literal, el contenido del literal se trata como la vía de acceso real. Si especifica *text-name* como una palabra definida por el usuario, el procesamiento depende de si se ha establecido la variable de entorno que corresponde a *text-name* . Si se establece la variable de entorno, el valor de la variable de entorno se utiliza como nombre de archivo, y posiblemente el nombre de vía de acceso, para el libro de copias.

Un *nombre-texto* se trata como una vía de acceso absoluta si se cumplen las tres condiciones siguientes:

- *nombre-biblioteca* no se utiliza.
- *nombre-texto* es un literal o una variable de entorno.
- El primer carácter es '/'.

Por ejemplo, esto se trata como una vía de acceso absoluta:

```
COPY "/mycpylib/mytext.cpy"
```

Si la variable de entorno que corresponde a *nombre-texto* no está establecida, la búsqueda del libro de copias utiliza los nombres siguientes:

1. *text-name* con sufijo .cpy
2. *text-name* con sufijo .cbl

3. *text-name* con sufijo .cob
4. *text-name* sin sufijo

Por ejemplo, `COPY MyCopy` busca en el orden siguiente:

1. MYCOPY.cpy (en todas las vías de acceso especificadas, tal como se ha descrito anteriormente)
2. MYCOPY.cbl (en todas las vías de acceso especificadas, como se ha descrito anteriormente)
3. MYCOPY.cob (en todas las vías de acceso especificadas, como se ha descrito anteriormente)
4. MYCOPY (en todas las vías de acceso especificadas, como se ha descrito anteriormente)

COBOL toma el valor predeterminado de *nombre-biblioteca* y *nombre-texto* en mayúsculas a menos que el nombre esté contenido en un literal ("MyCopy"). En este ejemplo, `MyCopy` no es lo mismo que `MYCOPY`. Si el nombre de archivo está en mayúsculas y minúsculas (como en MyCopy.cbl), defina *text-name* como un literal en la sentencia `COPY` .

-I opción

Para otros casos (cuando ni un *nombre-biblioteca* ni un *nombre-texto* indican la vía de acceso), la vía de acceso de búsqueda depende de la opción `-I` .

Para que `COPY A` sea equivalente a `COPY A OF MYLIB`, especifique `-I$MYLIB`.

Basándose en las reglas anteriores, se buscará en `COPY "/X/Y"` en el directorio raíz y se buscará en `COPY "X/Y"` en el directorio actual.

`COPY A OF SYSLIB` es equivalente a `COPY A`. La opción `-I` no afecta a las sentencias `COPY` que tienen cualificaciones de *nombre-biblioteca* explícitas además de las que tienen el nombre de biblioteca de `SYSLIB`.

Si se especifican *nombre-biblioteca* y *nombre-texto* , el compilador inserta un separador de vía de acceso (/) entre los dos valores si *nombre-biblioteca* no termina en /. Por ejemplo, `COPY MYCOPY OF MYLIB` con estos valores:

```
export MYCOPY=MYPDS(MYMEMBER)
export MYLIB=MYFILE
```

da como resultado `MYFILE/MYPDS(MYMEMBER)`.

Si especifica *text-name* como una palabra definida por el usuario, puede acceder a los archivos locales y también a los miembros PDS en z/OS sin cambiar el origen del sistema principal. Por ejemplo:

```
COPY mycopybook
```

En este ejemplo, si la variable de entorno *mycopybook* se establece en `h/mypds(mycopy)`:

- `h` se asigna al host específico.
- `mypds` es el nombre PDS de z/OS .
- `mycopy` es el nombre del miembro PDS.

Puede acceder a los archivos z/OS desde Linux utilizando NFS (Network File System), que le permite acceder a los archivos z/OS utilizando el nombre de vía de acceso de a Linux . Sin embargo, tenga en cuenta que NFS convierte el separador de vía de acceso en "." para seguir los convenios de denominación de z/OS . Para garantizar la correcta formación de nombres, tenga esto en cuenta al asignar valores a las variables de entorno. Por ejemplo, estos valores:

```
export MYCOPY=(MYMEMBER)
export MYLIB=M/MYFILE/MYPDS
```

no funcionan porque la vía de acceso resultante es:

```
M/MYFILE/MYPDS/(MYMEMBER)
```

que después de la conversión del separador de vía de acceso se convierte en:

```
M.MYFILE.MYPDS.(MYMEMBER)
```

DELETE sentencia
Esta sentencia de biblioteca fuente ampliada elimina las sentencias COBOL del programa fuente BASIS .

EJECT sentencia
Esta sentencia de dirección de compilador especifica que la siguiente sentencia fuente debe imprimirse en la parte superior de la página siguiente.

ENTER sentencia
La sentencia se trata como un comentario.

DirectivaEVALUATE
La directiva EVALUATE proporciona un método de varias ramificaciones para elegir las líneas fuente que se van a incluir en un grupo de compilación.

DirectivaIF
La directiva IF establece una compilación condicional unidireccional o bidireccional.

INSERT sentencia
Esta sentencia de biblioteca añade sentencias COBOL al programa fuente BASIS .

PROCESS (CBL) sentencia
Esta sentencia de direccionamiento de compilador, que puede colocar antes de la cabecera IDENTIFICATION DIVISION de un programa más externo, especifica las opciones de compilador que se van a utilizar durante la compilación del programa.

REPLACE sentencia
Esta sentencia se utiliza para sustituir el texto del programa fuente.

SKIP1/2/3 sentencia
Estas sentencias indican líneas que se deben omitir en el listado fuente.

TITLE sentencia
Esta sentencia especifica que debe imprimirse un título (cabecera) en la parte superior de cada página del listado fuente.

USE sentencia
La sentencia USE proporciona *declaratives* para especificar estos elementos:

- Procedimientos de manejo de errores: EXCEPTION/ERROR
- Procedimientos de manejo de etiquetas de usuario: LABEL
- Depuración de líneas y secciones: DEBUGGING

Tareas relacionadas
“Cambio de la cabecera de un listado de origen” en la página 4
“Compilación desde la línea de mandatos” en la página 240
“Especificación de opciones de compilador en la sentencia PROCESS (CBL)” en la página 242

Referencias relacionadas
“Opciones cob2” en la página 248
CALLINTERFACE (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia PROCESS (CBL) (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia *CONTROL (*CBL) (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia COPY (*COBOL for Linux en x86 Consulta de lenguaje*)
DEFINE directiva (*COBOL for Linux en x86 Consulta de lenguaje*)

Directiva EVALUATE (*COBOL for Linux en x86 Consulta de lenguaje*)
Directiva IF (*COBOL for Linux en x86 Consulta de lenguaje*)

Capítulo 15. Opciones de tiempo de ejecución

La tabla siguiente lista las opciones de tiempo de ejecución que están soportadas.

Tabla 32. ***Opciones de tiempo de ejecución***

Opción	Descripción	Valor predeterminado	Abreviatura
"COMPROBAR" en la página 315	Distintivos que comprueban errores	`CHECK(ON)`	`CH`
"DEPURAR" en la página 316	Especifica si las secciones de depuración COBOL especificadas por la declarativa `USE FOR DEBUGGING` están activas	`NODEBUG`	Ninguna
"RECUENTO de ERRORES" en la página 316	Especifica cuántas condiciones de gravedad 1 (nivel W) se pueden producir antes de que la unidad de ejecución termine de forma anómala	`ERRCOUNT(20)`	Ninguna
"ARCHIVOS" en la página 317	Especifica el sistema de archivos que se debe utilizar para los archivos para los que no se realiza ninguna selección explícita del sistema de archivos, ya sea a través de la cláusula `ASSIGN` o de una variable de entorno	`FILESYS(VSA)`	`FS(DB2|QSAM|RSD|SdU|SFS |STL|VSA|VSAM)`
"TRAP" en la página 318	Indica si COBOL intercepta excepciones	`TRAP(ON)`	Ninguna
"UPSI" en la página 318	Establece los ocho conmutadores `UPSI` activados o desactivados para las aplicaciones que utilizan rutinas COBOL	`UPSI(00000000)`	Ninguna

Especifique las opciones de tiempo de ejecución estableciendo la variable de entorno de tiempo de ejecución COBRTOPT.

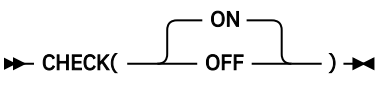
Tareas relacionadas
"Establecimiento de variables de entorno" en la página 229
"ejecutar programas" en la página 254

Referencias relacionadas
"Variables de entorno de ejecución" en la página 234

COMPROBAR

CHECK hace que se marquen los errores de comprobación. En COBOL, los rangos de índice, subíndice y modificación de referencia pueden provocar errores de comprobación.

Sintaxis de la opción CHECK



El valor predeterminado es: `CHECK(ON)`

La abreviatura es: CH

ON
Especifica que se realiza la comprobación de tiempo de ejecución

OFF
Especifica que no se realiza la comprobación de tiempo de ejecución

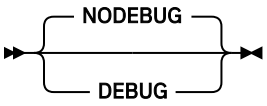
Nota de uso: CHECK(ON) no tiene ningún efecto si NOSSRANGE estaba en vigor durante la compilación.

Consideración sobre el rendimiento: Si ha compilado un programa COBOL con SSRANGEy no está probando o depurando una aplicación, el rendimiento mejora si especifica CHECK(OFF).

DEPURAR

DEBUG especifica si las secciones de depuración COBOL especificadas por la declarativa USE FOR DEBUGGING están activas.

Sintaxis de la opción DEBUG



El valor predeterminado es: NODEBUG

DEBUG
Activa las secciones de depuración

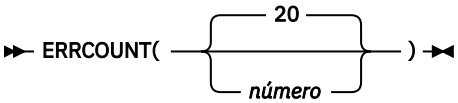
NODEBUG
Suprime las secciones de depuración

Consideración sobre el rendimiento: Para mejorar el rendimiento, utilice esta opción sólo durante la depuración.

RECUENTO de ERRORES

ERRCOUNT indica cuántos mensajes de aviso se pueden producir antes de que la unidad de ejecución termine de forma anómala.

Sintaxis de la opción ERRCOUNT



DEFAULT: ERRCOUNT(20)

número, si es positivo, es el número de mensajes de aviso que se pueden producir mientras se ejecuta la unidad de ejecución. Si el número de mensajes de aviso excede *número*, la unidad de ejecución termina de forma anómala.

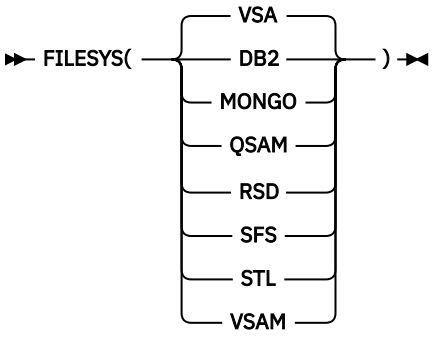
número, si es 0, indica que se puede producir un número ilimitado de mensajes de aviso. *número* no puede ser negativo.

Cualquier mensaje debido a una condición que tenga una gravedad mayor que un aviso da como resultado la terminación de la unidad de ejecución independientemente del valor de la opción ERRCOUNT .

ARCHIVOS

FILESYS especifica el sistema de archivos que se va a utilizar para los archivos para los que no se ha especificado ningún sistema de archivos explícito mediante una cláusula ASSIGN o una variable de entorno. La opción se aplica a archivos secuenciales, relativos e indexados. No se aplica a los archivos secuenciales de línea, para los que el sistema de archivos debe especificarse como, o de forma predeterminada, LSQ (línea secuencial).

Sintaxis de la opción FILESYS



El valor predeterminado es: FILESYS(VSA)

La abreviatura es: FS(DB2|MON|QSA|RSD|SFS|STL|VSA)

DB2
El sistema de archivos es una base de datos relacional Db2 .

MONGO
El sistema de archivos es una base de datos MongoDB .

QSAM
El sistema de archivos es compatible con los archivos QSAM del sistema principal.

RSD
El sistema de archivos está delimitado secuencialmente por registros.

SFS
El sistema de archivos es CICS Structured File Server.

STL
El sistema de archivos es el sistema de archivos de idioma estándar.

VSA o VSAM
VSA o VSAM (método de acceso de almacenamiento virtual) implica el sistema de archivos SFS o STL .

Si el nombre de archivo del sistema empieza por el valor `/.:/cics/sfs`, esta subopción implica el sistema de archivos SFS. De lo contrario, implica el sistema de archivos STL .

Conceptos relacionados
“Sistemas de archivos” en la página 122
“Organización de archivo secuencial de línea” en la página 128

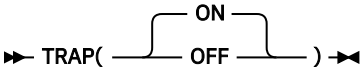
Tareas relacionadas
“Identificación de archivos” en la página 117

Referencias relacionadas
“Prioridad de la determinación del sistema de archivos” en la página 121
“Variables de entorno de ejecución” en la página 234
Cláusula ASSIGN (*COBOL for Linux en x86 Consulta de lenguaje*)

TRAP

TRAP indica si COBOL intercepta excepciones.

Sintaxis de la opción TRAP



El valor predeterminado es: TRAP(ON)

Si TRAP(OFF) está en vigor y no proporciona su propio manejador de condiciones de excepción para manejar condiciones excepcionales, las condiciones dan como resultado una acción predeterminada por parte del sistema operativo. Por ejemplo, si el programa intenta almacenar en una ubicación no permitida, la acción del sistema predeterminada es emitir un mensaje y terminar el proceso.

ON
Activa la intercepción COBOL de excepciones

OFF
Desactiva la intercepción COBOL de excepciones

Notas de uso

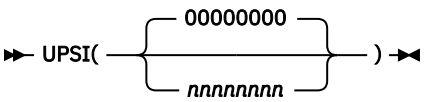
- Utilice TRAP(OFF) sólo cuando necesite analizar una excepción de programa antes de que COBOL la maneje.
- Cuando especifica TRAP(OFF) en un entorno que no esCICS , no se establece ningún manejador de excepciones.
- La ejecución con TRAP(OFF) (para fines de diagnóstico de excepciones) puede provocar muchos efectos secundarios, porque COBOL requiere TRAP(ON). Cuando ejecuta con TRAP(OFF), puede obtener efectos secundarios incluso si no encuentra una condición de software generado, comprobación de programa o terminación anómala. Si encuentra una comprobación de programa o una terminación anómala con TRAP(OFF) en vigor, se pueden producir los siguientes efectos secundarios:
  - Los recursos obtenidos por COBOL no se liberan.
  - Los archivos abiertos por COBOL no se cierran, por lo que es posible que se pierdan registros.
  - No se generan mensajes ni salidas de volcado.

  La unidad de ejecución termina de forma anómala si se dan estas condiciones.

UPSI

UPSI establece los ocho conmutadores UPSI activados o desactivados para las aplicaciones que utilizan rutinas COBOL.

Sintaxis de la opción UPSI



El valor predeterminado es: UPSI(00000000)

Cada *n* representa un conmutador UPSI (entre 0 y 7); el *n* más a la izquierda representa el primer conmutador. Cada *n* puede ser 0 (desactivado) o 1 (activado).

Capítulo 16. depuración

Puede elegir entre dos enfoques diferentes para determinar la causa de los problemas en el comportamiento de la aplicación: depuración de lenguaje fuente o depuración interactiva.

Para la depuración de lenguaje fuente, COBOL proporciona varios elementos de lenguaje, opciones de compilador y salidas de listado que facilitan la depuración.

Para la depuración interactiva, puede utilizar IBM Debug for Linux en x86.

Tareas relacionadas
“Depuración con lenguaje fuente” en la página 319
“Depuración utilizando opciones de compilador” en la página 323
“Depuración utilizando IBM Debug for Linux en x86” en la página 328
“Obtención de listados” en la página 382
“Depuración con mensajes que tienen información de desplazamiento” en la página 395

“Depuración de rutinas de ensamblador” en la página 395

Depuración con lenguaje fuente

Puede utilizar varias características de lenguaje COBOL para determinar la causa de una anomalía en un programa.

Acerca de esta tarea

Si un programa anómalo forma parte de una aplicación grande que ya está en producción (excluyendo actualizaciones de origen), escriba un pequeño caso de prueba para simular la parte anómala del programa. Características de depuración de código en el caso de prueba para ayudar a detectar estos problemas:

- Errores en la lógica del programa
- Errores de entrada-salida
- Discrepancias de tipos de datos
- Datos no inicializados
- Problemas con los procedimientos

Tareas relacionadas
“Rastreo de la lógica del programa” en la página 319
“Búsqueda y manejo de errores de entrada-salida” en la página 320
“Validación de datos” en la página 321
“Mover, inicializar o establecer datos no inicializados” en la página 321
“Generación de información sobre procedimientos” en la página 321

Referencias relacionadas
Depuración de lenguaje fuente (*COBOL for Linux en x86 Consulta de lenguaje*)

Rastreo de la lógica del programa

Rastree la lógica del programa añadiendo sentencias `DISPLAY` .

Acerca de esta tarea

Por ejemplo, si determina que el problema está en una sentencia `EVALUATE` o en un conjunto de sentencias `IF` anidadas, utilice sentencias `DISPLAY` en cada vía de acceso para ver el flujo lógico. Si determina que el cálculo de un valor numérico está causando el problema, utilice sentencias `DISPLAY` para comprobar el valor de algunos resultados provisionales.

Si utiliza terminadores de ámbito explícitos para finalizar sentencias en el programa, la lógica es más aparente y, por lo tanto, más fácil de rastrear.

Para determinar si una rutina determinada se ha iniciado y ha finalizado, puede insertar un código como este en el programa:

```
DISPLAY "ENTER CHECK PROCEDURE"
    .
    .  (checking procedure routine)
    .
DISPLAY "FINISHED CHECK PROCEDURE"
```

Después de estar seguro de que la rutina funciona correctamente, inhabilite las sentencias DISPLAY de una de estas dos maneras:

- Ponga un asterisco en la columna 7 de cada línea de sentencia DISPLAY para convertirla en una línea de comentario.
- Coloque un D en la columna 7 de cada sentencia DISPLAY para convertirla en una línea de comentario. Cuando desee reactivar estas sentencias, incluya una cláusula WITH DEBUGGING MODE en ENVIRONMENT DIVISION; se ignorará D en la columna 7 y se implementarán las sentencias DISPLAY .

Antes de poner el programa en producción, suprima o inhabilite las ayudas de depuración que ha utilizado y vuelva a compilar el programa. El programa se ejecutará de forma más eficiente y utilizará menos almacenamiento.

Conceptos relacionados
“Terminadores de ámbito” en la página 17

Referencias relacionadas
Sentencia DISPLAY (*COBOL for Linux en x86 Consulta de lenguaje*)

Búsqueda y manejo de errores de entrada-salida

Las claves de estado de archivo pueden ayudarle a determinar si los errores de programa se deben a errores de entrada-salida que se producen en el soporte de almacenamiento.

Acerca de esta tarea

Para utilizar claves de estado de archivo en la depuración, compruebe si hay un valor distinto de cero en la clave de estado después de cada sentencia de entrada-salida. Si el valor es distinto de cero (tal como se indica en un mensaje de error), consulte la codificación de los procedimientos de entrada-salida en el programa. También puede incluir procedimientos para corregir el error basándose en el valor de la clave de estado.

Si determina que un problema reside en un procedimiento de entrada-salida, incluya la declarativa USE EXCEPTION/ERROR para ayudar a depurar el problema. A continuación, cuando un archivo no se puede abrir, se realiza la declaración EXCEPTION/ERROR adecuada. La declaración adecuada puede ser una específica para el archivo o una proporcionada para los atributos abiertos INPUT, OUTPUT, I-Oo EXTEND.

Codifique cada sentencia USE AFTER STANDARD ERROR en una sección que sigue a la palabra clave DECLARATIVES en PROCEDURE DIVISION.

Tareas relacionadas
“Declaración de ERROR de codificación” en la página 180
“Utilización de claves de estado de archivo” en la página 181

Referencias relacionadas
Clave de estado de archivo (*COBOL for Linux en x86 Consulta de lenguaje*)

Validación de datos

Si sospecha que el programa está intentando realizar la aritmética en datos no numéricos o está recibiendo el tipo incorrecto de datos en un registro de entrada, utilice la prueba de clase (la condición de clase) para validar el tipo de datos.

Acerca de esta tarea

Puede utilizar la prueba de clase para comprobar si el contenido de un elemento de datos es ALPHABETIC, ALPHABETIC-LOWER, ALPHABETIC-UPPER, DBCS, KANJIo NUMERIC. Si el elemento de datos se describe implícita o explícitamente como USAGE NATIONAL, la prueba de clase comprueba la representación de caracteres nacionales de los caracteres asociados con la clase de caracteres especificada.

Tareas relacionadas
"Codificación de expresiones condicionales" en la página 87
"Prueba de caracteres DBCS válidos" en la página 212

Referencias relacionadas
Condición de clase (*COBOL for Linux en x86 Consulta de lenguaje*)

Mover, inicializar o establecer datos no inicializados

Utilice una sentencia INITIALIZE o SET para inicializar una tabla o un elemento de datos cuando sospeche que un problema puede estar causado por datos residuales en esos campos.

Acerca de esta tarea

Si el problema se produce de forma intermitente y no siempre con los mismos datos, podría ser que un conmutador no se inicializara, pero generalmente se establece en el valor correcto (0 o 1) por casualidad. Utilizando una sentencia SET para asegurarse de que el conmutador se ha inicializado, puede determinar que el conmutador no inicializado es la causa del problema o eliminarlo como posible causa.

Referencias relacionadas
sentencia INITIALIZE (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia SET (*COBOL for Linux en x86 Consulta de lenguaje*)

Generación de información sobre procedimientos

Genere información sobre el programa o caso de prueba y cómo se ejecuta codificando la declaración USE FOR DEBUGGING . Esta declaración le permite incluir sentencias en el programa e indicar cuándo deben realizarse al ejecutar el programa.

Acerca de esta tarea

Por ejemplo, para determinar cuántas veces se ejecuta un procedimiento, puede incluir un procedimiento de depuración en la declarativa USE FOR DEBUGGING y utilizar un contador para realizar un seguimiento del número de veces que el control pasa a dicho procedimiento. Puede utilizar la técnica de contador para comprobar elementos como los siguientes:

- Cuántas veces se ejecuta una sentencia PERFORM y, por lo tanto, si se está utilizando una rutina determinada y si la estructura de control es correcta
- Cuántas veces se ejecuta un bucle y, por lo tanto, si el bucle se está ejecutando y si el número del bucle es preciso

Puede utilizar líneas de depuración o sentencias de depuración o ambas en el programa.

Las *líneas de depuración* son sentencias identificadas por un D en la columna 7. Para activar las líneas de depuración en el programa, codifique la cláusula WITH DEBUGGING MODE en la línea SOURCE-COMPUTER en ENVIRONMENT DIVISION. De lo contrario, las líneas de depuración se tratan como comentarios.

Las *sentencias de depuración* son las sentencias codificadas en la sección DECLARATIVES de PROCEDURE DIVISION. Codifique cada USE FOR DEBUGGING declarativo en una sección separada. Codifique las sentencias de depuración como se indica a continuación:

- Sólo en una sección DECLARATIVES .
- Siguiendo la cabecera USE FOR DEBUGGING.
- Sólo en el programa más externo; no son válidos en programas anidados. Las sentencias de depuración tampoco se desencadenan nunca mediante procedimientos contenidos en programas anidados.

Para utilizar sentencias de depuración en el programa, debe incluir la cláusula WITH DEBUGGING MODE y utilizar la opción de tiempo de ejecución DEBUG .

Restricciones de opciones:

- Los declarativos USE FOR DEBUGGING , si se ha especificado la cláusula WITH DEBUGGING MODE , se excluyen mutuamente con la opción de compilador TEST . Si están presentes los declarativos USE FOR DEBUGGING y la cláusula WITH DEBUGGING MODE , se cancela la opción TEST .

"Ejemplo: USE FOR DEBUGGING" en la página 322

Referencias relacionadas
párrafo SOURCE-COMPUTER (*COBOL for Linux en x86 Consulta de lenguaje*)
Depuración de líneas (*COBOL for Linux en x86 Consulta de lenguaje*)
Secciones de depuración (*COBOL for Linux en x86 Consulta de lenguaje*)
DEBUGGING declarativo (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: USE FOR DEBUGGING

Este ejemplo muestra el tipo de sentencias necesarias para utilizar una sentencia DISPLAY y una declarativa USE FOR DEBUGGING para probar un programa.

La sentencia DISPLAY graba información en el terminal o en un archivode salida. La declarativa USE FOR DEBUGGING se utiliza con un contador para mostrar cuántas veces se ejecuta una rutina.

```
Environment Division.
. . .
Data Division.
. . .
Working-Storage Section.
. . . (other entries your program needs)
01  Trace-Msg    PIC X(30) Value "  Trace for Procedure-Name : ".
01  Total        PIC 9(9)  Value 1.
. . .
Procedure Division.
Declaratives.
Debug-Declaratives Section.
    Use For Debugging On Some-Routine.
Debug-Declaratives-Paragraph.
    Display Trace-Msg, Debug-Name, Total.
End Declaratives.

Main-Program Section.
    . . . (source program statements)
    Perform Some-Routine.
    . . . (source program statements)
    Stop Run.
Some-Routine.
    . . . (whatever statements you need in this paragraph)
    Add 1 To Total.
Some-Routine-End.
```

La sentencia DISPLAY de DECLARATIVES SECTION emite este mensaje cada vez que se ejecuta el procedimiento Some-Routine :

```
Trace For Procedure-Name : Some-Routine 22
```

El número al final del mensaje, 22, es el valor acumulado en el elemento de datos `Total`; indica el número de veces que se ha ejecutado `Some-Routine` . Las sentencias de la declarativa de depuración se realizan antes de que se ejecute el procedimiento con nombre.

También puede utilizar la sentencia `DISPLAY` para rastrear la ejecución del programa y mostrar el flujo a través del programa. Para ello, descarte `Total` de la sentencia `DISPLAY` y cambie la declaración `USE FOR DEBUGGING` en `DECLARATIVES SECTION` por:

```
USE FOR DEBUGGING ON ALL PROCEDURES.
```

Como resultado, se visualiza un mensaje antes de que se ejecute cada procedimiento de no depuración en el programa más externo.

Depuración utilizando opciones de compilador

Puede utilizar determinadas opciones de compilador para ayudarle a encontrar errores en el programa, encontrar varios elementos en el programa, obtener listados y preparar el programa para la depuración.

Acerca de esta tarea

Puede encontrar los errores siguientes utilizando opciones de compilador (las opciones se muestran entre paréntesis):

- Errores de sintaxis como, por ejemplo, nombres de datos duplicados (`NOCOMPILE`)
- Secciones que faltan (`SEQUENCE`)
- Valores de subíndice no válidos (`SSRANGE`)

Puede encontrar los siguientes elementos en el programa utilizando las opciones del compilador:

- Mensajes de error y ubicaciones de los errores asociados (`FLAG`)
- Definiciones de entidad de programa y referencias (`XREF`)
- Elementos de datos en `DATA DIVISION` (`MAP`)
- Referencias de sentencia (`VBREF`)

Puede obtener una copia del origen (`SOURCE`) o un listado del código generado (`LIST`).

Puede preparar el programa para la depuración utilizando la opción de compilador `TEST` .

Tareas relacionadas
“Búsqueda de errores de codificación” en la página 323
“Búsqueda de problemas de secuencia de línea” en la página 324
“Comprobación de rangos válidos” en la página 324
“Selección del nivel de error que se va a diagnosticar” en la página 325
“Búsqueda de definiciones y referencias de entidad de programa” en la página 327
“Listado de elementos de datos” en la página 327
“Obtención de listados” en la página 382

Referencias relacionadas
“opciones de compilador” en la página 265

Búsqueda de errores de codificación

Utilice la opción `NOCOMPILE` para compilar condicionalmente o sólo para comprobar la sintaxis. Cuando se utiliza con la opción `SOURCE` , `NOCOMPILE` genera un listado que le ayudará a encontrar errores de codificación como, por ejemplo, definiciones que faltan, elementos de datos definidos incorrectamente y nombres de datos duplicados.

Acerca de esta tarea

Comprobación sólo de sintaxis: Para comprobar sólo la sintaxis del programa y no producir código de objeto, utilice NOCOMPILE sin una subopción. Si también especifica la opción SOURCE , el compilador genera un listado.

Cuando especifica NOCOMPILE, se suprimen varias opciones de compilador. Consulte la referencia relacionada a continuación sobre la opción COMPILE para obtener más detalles.

Compilación condicional: Para compilar condicionalmente, utilice NOCOMPILE(*x*), donde *x* es uno de los niveles de gravedad de los errores. El programa se compila si todos los errores tienen una gravedad inferior a *x*. Los niveles de gravedad que puede utilizar, de mayor a menor, son S (grave), E (error) y W (aviso).

Si se produce un error de nivel *x* o superior, la compilación se detiene y sólo se comprueba la sintaxis del programa.

Referencias relacionadas
"COMPILAR" en la página 276

Búsqueda de problemas de secuencia de línea

Utilice la opción de compilador SEQUENCE para buscar sentencias que están fuera de secuencia. Las interrupciones en la secuencia indican que se ha movido o suprimido una sección de un programa fuente.

Acerca de esta tarea

Cuando se utiliza SEQUENCE, el compilador comprueba los números de sentencia de origen para determinar si están en secuencia ascendente. Se colocan dos asteriscos junto a los números de sentencia que están fuera de secuencia. El número total de estas sentencias se imprime como la primera línea en los diagnósticos después del listado fuente.

Referencias relacionadas
"SECUENCIA" en la página 295

Comprobación de rangos válidos

Utilice la opción de compilador SSRANGE para comprobar si las direcciones se encuentran dentro de los rangos adecuados.

Acerca de esta tarea

SSRANGE hace que se comprueban las direcciones siguientes:

- Referencias de datos con subíndice o indexados: ¿Es la dirección efectiva del elemento required dentro del límite máximo de la tabla especificada?
- Referencias de datos de longitud variable (una referencia a un elemento de datos que contiene una cláusula OCCURS DEPENDING ON): ¿Es la longitud real positiva y está dentro de la longitud máxima definida para el elemento de datos de grupo?
- Referencias de datos modificadas por referencia: ¿El desplazamiento y la longitud son positivos? ¿Es la suma del desplazamiento y la longitud dentro de la longitud máxima para el elemento de datos?

Si la opción SSRANGE está en vigor, la comprobación se realiza en tiempo de ejecución si se cumplen las dos condiciones siguientes:

- Se realiza la sentencia COBOL que contiene el elemento de datos indexado, subíndice, de longitud variable o modificado por referencia.
- La opción de tiempo de ejecución CHECK es ON.

Si una dirección efectiva está fuera del rango del elemento de datos que contiene los datos referenciados, se genera un mensaje de error y el programa se detiene. El mensaje identifica la tabla o identificador al

que se hace referencia y el número de línea donde se ha producido el error. Se proporciona información adicional en función del tipo de referencia que ha causado el error.

Si todos los subíndices, índices y modificadores de referencia de una referencia de datos determinada son literales y dan como resultado una referencia fuera del elemento de datos, el error se diagnostica durante la compilación independientemente del valor de la opción SSRANGE .

Consideración sobre el rendimiento: SSRANGE puede degradar un poco el rendimiento debido a la sobrecarga adicional para comprobar cada elemento subíndice o indexado.

Referencias relacionadas
“SRANGE” en la página 299
“Opciones de compilador relacionadas con el rendimiento” en la página 534

Selección del nivel de error que se va a diagnosticar

Utilice la opción de compilador FLAG para especificar el nivel de error que se va a diagnosticar durante la compilación y para indicar si los mensajes de error se van a incluir en el listado. Utilice `FLAG(I)` o `FLAG(I,I)` para que se le notifique de todos los errores.

Acerca de esta tarea

Especifique como primer parámetro el nivel de gravedad más bajo de los mensajes de error de sintaxis que deben emitirse. Opcionalmente, especifique el segundo parámetro como el nivel más bajo de los mensajes de error de sintaxis que se van a incluir en el listado de origen. Este nivel de gravedad debe ser el mismo o mayor que el nivel del primer parámetro. Si especifica ambos parámetros, también debe especificar la opción de compilador SOURCE .

Tabla 33. ***Niveles de gravedad de los mensajes del compilador***

Nivel de gravedad	**Mensajes resultantes**
U (irrecuperable)	Sólo mensajes U
S (grave)	Todos los mensajes S y U
E (error)	Todos los mensajes E, S y U
W (aviso)	Todos los mensajes W, E, S y U
I (informativo)	Todos los mensajes

Cuando se especifica el segundo parámetro, cada mensaje de error de sintaxis (excepto un mensaje de nivel U) se incorpora en el listado fuente en el punto en el que el compilador tenía suficiente información para detectar dicho error. Todos los mensajes incorporados (excepto los emitidos por la fase del compilador de bibliotecas) siguen directamente la sentencia a la que hacen referencia. El número de la sentencia que ha tenido el error también se incluye con el mensaje. Los mensajes incorporados se repiten con el resto de los mensajes de diagnóstico al final del listado fuente.

Cuando se especifica la opción de compilador NOSOURCE , los mensajes de error de sintaxis sólo se incluyen al final del listado. Los mensajes para errores irrecuperables no están incorporados en el listado fuente, porque un error de esta gravedad termina la compilación.

“Ejemplo: mensajes incorporados” en la página 326

Tareas relacionadas
“Generación de una lista de mensajes del compilador” en la página 246

Referencias relacionadas
“Códigos de gravedad para mensajes de diagnóstico del compilador” en la página 245
“Mensajes y listados de errores detectados por el compilador” en la página 246
“BANDERA” en la página 284

Ejemplo: mensajes incorporados

El ejemplo siguiente muestra los mensajes incorporados generados especificando un segundo parámetro en la opción FLAG . Algunos mensajes del resumen se aplican a más de una sentencia COBOL.

```
   LineID  PL SL  ----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7-|--+----8  Map
and Cross Reference
 ...                                                                                          |
   000977              /                                                                      |
   000978              ***************************************************************        |
   000979              ***          I N I T I A L I Z E   P A R A G R A P H          **        |
   000980              ***  Open files. Accept date, time and format header lines.    **        |
   000981         IA4690***   Load location-table.                                    **        |
   000982              ***************************************************************        |
   000983                 100-initialize-paragraph.                                            |
   000984                   move spaces to ws-transaction-record                               | IMP
339
   000985                   move spaces to ws-commuter-record                                  | IMP
315
   000986                   move zeroes to commuter-zipcode                                    | IMP
326
   000987                   move zeroes to commuter-home-phone                                 | IMP
327
   000988                   move zeroes to commuter-work-phone                                 | IMP
328
   000989                   move zeroes to commuter-update-date                                | IMP
332
   000990                   open input update-transaction-file                                 | 203
 ==000990==> IGYPS2052-S An error was found in the definition of file "LOCATION-FILE".  The       |
                          reference to this file was discarded.                               |
   000991                         location-file                                                | 192
   000992                         i-o commuter-file                                            | 180
   000993                         output print-file                                            | 216
   000994                   if loccode-file-status not = "00" or                               | 248
   000995                      update-file-status not = "00" or                                | 247
   000996                      updprint-file-status not = "00"                                 | 249
   000997      1               display "Open Error ..."                                        |
   000998      1               display "  Location File Status = " loccode-file-status         | 248
   000999      1               display "  Update   File Status = " update-file-status          | 247
   001000      1               display "  Print    File Status = " updprint-file-status       | 249
   001001      1             perform 900-abnormal-termination                                  | 1433
   001002                   end-if                                                             |
   001003         IA4760     if commuter-file-status not = "00" and not = "97"                  | 240
   001004      1               display "100-OPEN"                                              |
   001005      1               move 100 to comp-code                                           | 230
   001006      1               perform 500-stl-error                                           | 1387
   001007      1               display "Commuter File Status (OPEN) = "                        |
   001008      1                       commuter-file-status                                    | 240
   001009      1               perform 900-abnormal-termination                                | 1433
   001010         IA4790     end-if                                                             |
   001011                   accept ws-date from date                                           | UND
 ==001011==> IGYPS2121-S "WS-DATE" was not defined as a data-name.  The statement was discarded.  |
   001012         IA4810     move corr ws-date to header-date                                   | UND
463
 ==001012==> IGYPS2121-S "WS-DATE" was not defined as a data-name.  The statement was discarded.  |
   001013                   accept ws-time from time                                           | UND
 ==001013==> IGYPS2121-S "WS-TIME" was not defined as a data-name.  The statement was discarded.  |
   001014         IA4830     move corr ws-time to header-time                                   | UND
457
 ==001014==> IGYPS2121-S "WS-TIME" was not defined as a data-name.  The statement was discarded.  |
   001015         IA4840     read location-file                                                 | 192
 ...
 LineID  Message code  Message text
      192  IGYDS1050-E   File "LOCATION-FILE" contained no data record descriptions.
                         The file definition was discarded.
      899  IGYPS2052-S   An error was found in the definition of file "LOCATION-FILE".
                         The reference to this file was discarded.
                         Same message on line:    990
     1011  IGYPS2121-S   "WS-DATE" was not defined as a data-name.  The statement was discarded.
                         Same message on line:   1012
     1013  IGYPS2121-S   "WS-TIME" was not defined as a data-name.  The statement was discarded.
                         Same message on line:   1014
     1015  IGYPS2053-S   An error was found in the definition of file "LOCATION-FILE".
                         This input/output statement was discarded.
                         Same message on line:   1027
     1026  IGYPS2121-S   "LOC-CODE" was not defined as a data-name.  The statement was discarded.
     1209  IGYPS2121-S   "COMMUTER-SHIFT" was not defined as a data-name.  The statement was discarded.
                         Same message on line:   1230
     1210  IGYPS2121-S   "COMMUTER-HOME-CODE" was not defined as a data-name.  The statement was
```

```
discarded.
                        Same message on line:   1231
     1212  IGYPS2121-S   "COMMUTER-NAME" was not defined as a data-name.  The statement was discarded.
                        Same message on line:   1233
     1213  IGYPS2121-S   "COMMUTER-INITIALS" was not defined as a data-name.  The statement was
discarded.
                        Same message on line:   1234
     1223  IGYPS2121-S   "WS-NUMERIC-DATE" was not defined as a data-name.  The statement was discarded.
 Messages    Total    Informational    Warning    Error    Severe    Terminating
 Printed:       19                                    1        18
 * Statistics for COBOL program FLAGOUT:
 *    Source records = 1755
 *    Data Division statements = 279
 *    Procedure Division statements = 479
 Locale = en_US.ISO8859-1                    (1)
 End of compilation 1,  program FLAGOUT,  highest severity: Severe.
 Return code 12
```

(1)
El entorno local que ha utilizado el compilador

Búsqueda de definiciones y referencias de entidad de programa

Utilice la opción de compilador XREF(FULL) para averiguar dónde se define y se hace referencia a un nombre de datos, un nombre de procedimiento o un nombre de programa. Utilícelo también para generar una referencia cruzada de sentencias COPY o BASIS a los conjuntos de datos de los que se han obtenido los libros de copias.

Acerca de esta tarea

Una referencia cruzada ordenada incluye el número de línea donde se ha definido el nombre de datos, el nombre de procedimiento o el nombre de programa y los números de línea de todas las referencias a él.

Para incluir sólo los elementos de datos a los que se hace referencia explícitamente, utilice la opción XREF(SHORT) .

Utilice las opciones XREF (FULL o SHORT) y SOURCE para imprimir una referencia cruzada modificada a la derecha del listado fuente. Esta referencia cruzada incorporada muestra el número de línea donde se ha definido el nombre de datos o el nombre de procedimiento.

Para obtener más detalles, consulte la referencia relacionada sobre la opción de compilador XREF .

“Ejemplo: Salida XREF: referencias cruzadas de nombre de datos” en la página 391
“Ejemplo: Salida XREF: referencias cruzadas de nombre de programa” en la página 392
“Ejemplo: Salida XREF: COPY/BASE referencias cruzadas” en la página 392
“Ejemplo: Salida XREF: referencia cruzada incorporada” en la página 393

Tareas relacionadas
“Obtención de listados” en la página 382

Referencias relacionadas
“XREF” en la página 306

Listado de elementos de datos

Utilice la opción de compilador MAP para crear un listado de los elementos DATA DIVISION y todos los elementos declarados implícitamente.

Acerca de esta tarea

Cuando se especifica la opción MAP , se genera un resumen de MAP incorporado que contiene información de MAP condensada a la derecha de la definiciónde datos de origen COBOL.Cuando los datos de XREF y un resumen de MAP incorporado están en la misma línea, el resumen incorporado se imprime en primer lugar.

Puede seleccionar o inhibir partes del resumen de MAP listado e incorporado MAP utilizando sentencias *CONTROL MAP|NOMAP (o *CBL MAP|NOMAP) en todo el origen. Por ejemplo:

```
*CONTROL NOMAP
    01  A
    02  B
*CONTROL MAP
```

"Ejemplo: salida MAP" en la página 387

Tareas relacionadas
"Obtención de listados" en la página 382

Referencias relacionadas
"MAP" en la página 288

Depuración utilizando IBM Debug for Linux en x86

Utilice IBM Debug for Linux on x86, el depurador de nivel de fuente interactivo que se suministra con COBOL para Linux para depurar los programas COBOL.

IBM Debug for Linux en x86 visión general

IBM Debug for Linux on x86 es un depurador de nivel de fuente interactivo. Funciona en estaciones de trabajo basadas en Windowsy Linuxconectadas de forma remota a un motor de depurador que se ejecuta en Linux en x86. IBM Depuración para Linux en x86 le permite depurar programas escritos en COBOL.

El depurador muestra los archivos fuente de la aplicación y los elementos de dichos archivos fuente. Puede realizar un solo paso, recorrer paso a paso, recorrer paso a paso o detener la ejecución en una línea o condición especificada. Al controlar la ejecución, puede supervisar variables, registros, memoria, pilas de llamadas y otros elementos.

IBM Debug for Linux on x86 está habilitado para Internet Protocol Versión 6 (IPv6).

instalación

Aprenda a instalar los componentes de IBM Debug for Linux en x86.

Componentes del depurador

IBM Debug for Linux en x86 utiliza un modelo de cliente/servidor compuesto de dos componentes:

- El motor de depuración (`irmtdbgc`), que es un componente de servidor instalado en una máquina Linux en x86 .
- El cliente de depuración, Remote Debug Eclipse User Interface (repositoriop2), que está disponible como un conjunto de características de Eclipse que amplía una instancia de Eclipse existente y se puede instalar en una estación de trabajo Linux en x86 o Ventanas 10.

Instalación del motor de depuración de Linux en x86

El motor de depuración de IBM Debug for Linux en x86 se instala de forma predeterminada cuando se utiliza el programa de utilidad de instalación predeterminado que se proporciona con el producto. Consulte la Guía de instalación para obtener más información sobre cómo instalar el compilador y el motor de depuración.

Instalación del cliente de depuración de Linux en x86 o Windows 10

El cliente de depuración, Remote Debug Eclipse User Interface (repositoriop2), está disponible como un conjunto de características de Eclipse que amplían una instancia de Eclipse existente.

Para obtener más información sobre cómo descargar el repositorio de p2 e instalar las características, consulte Instalación de la interfaz de usuario de IBM Debug for Linux on x86 Remote Debug Eclipse.

Funciones de accesibilidad

Las funciones de accesibilidad ayudan a un usuario que tiene una discapacidad física, como por ejemplo movilidad restringida o visión limitada, a utilizar correctamente los productos de software.

IBM Depuración para Linux en x86 ofrece las siguientes características de accesibilidad:

- Indicadores de foco visual mediante cursores en objetos editables y botones resaltados, elementos de menú y otras selecciones.
- Ayuda contextual para botones y otras selecciones.
- Completar la habilitación de la tecnología de asistencia en asistentes y recuadros de diálogo.
- Mensajes, cuadros de diálogo y asistentes que persisten hasta que los cierra.
- Documentación que incluye descripciones de imagen contextual y cabeceras de tabla marcadas.
- Navegación por el teclado de la interfaz de usuario.

  Nota: Mientras se encuentra en un Eclipse IDE, puede abrir el menú contextual de la barra de marcadores del editor pulsando Ctrl+F10.

Navegación por la interfaz de usuario utilizando el teclado

Se puede navegar por la interfaz de usuario utilizando el teclado. La tecla de tabulación se utiliza para iterar a través de los controles en un ámbito determinado (por ejemplo, un diálogo o una vista y sus iconos relacionados). Para ir a los controles principales de la interfaz de usuario o para salir de las vistas que utilizan la tecla Tabulador (como los editores), utilice Ctrl + Tabulador.

Menús

Se puede acceder a los menús utilizando el teclado de las siguientes maneras:

- F10 accede a los menús de la barra de menús principal.
- Shift+F10 abre el menú contextual para la vista actual.

  Nota: Este acceso directo depende del gestor de ventanas. En la mayoría de los casos, es Shift+F10.
- Ctrl+F10 abre el menú desplegable (si lo hay) para la vista actual. Para los editores, Ctrl+F10 abrirá el menú de la barra de marcadores a la izquierda del área del editor.
- Alt + mnemónico activará el menú principal para una entrada en particular (por ejemplo, Alt + W abrirá el menú Ventana).
- Microsoft Windows solamente: al pulsar Alt se pondrá el foco en la barra de menús.

Controles

Los mnemónicos se asignan a la mayoría de las etiquetas de control (por ejemplo, botones, recuadros de selección y botones de selección) en recuadros de diálogo, páginas de preferencias y páginas de propiedades. Para acceder al control asociado a una etiqueta, utilice la tecla Alt junto con la letra subrayada en la etiqueta.

Contexto de navegación

El contexto de navegación se guarda para los diálogos de preferencias y propiedades. La página seleccionada para el diálogo de preferencias y propiedades se guarda entre las invocaciones del diálogo, pero no se guarda entre las invocaciones de la interfaz de usuario.

Ciclismo de editores, vistas y perspectivas

Para conmutar entre editores, vistas y perspectivas, la interfaz de usuario proporciona una función cíclica que se invoca mediante Ctrl y una tecla de función. Todas estas funciones de ciclismo recuerdan lo último que se ha seleccionado para permitir un ciclo rápido de ida y vuelta entre dos elementos. Las funciones cíclicas son:

- Ctrl+F6 -Ciclo al editor
- Ctrl+F7 -Ciclo para ver
- Ctrl+F8 -Ciclo a perspectiva

Además, se puede utilizar Ctrl + E para activar el desplegable del editor-y Ctrl + PageUp y Ctrl + PageDown se pueden utilizar para conmutar entre editores abiertos.

Asistencia clave

Muchas de las acciones de la interfaz de usuario tienen enlaces de teclado asignados. Para acceder a la lista de enlaces de teclado disponibles, seleccione **Ayuda > Asistencia de teclas** en el menú principal.

Sistema de ayuda

Puede navegar por el sistema de ayuda utilizando el teclado utilizando las siguientes combinaciones de teclas:

- Al pulsar el tabulador dentro de un marco (página), se conduce al siguiente enlace, botón o nodo de tema.
- Para expandir un nodo de árbol, pulse la flecha derecha. Para contraer un nodo de árbol, pulse la flecha izquierda.
- Para pasar al siguiente nodo de tema, pulse la flecha abajo o el tabulador.
- Para desplazarse al nodo de tema anterior, pulse Flecha arriba o Mayús + Tabulador.
- Para visualizar el tema seleccionado, pulse Intro.
- Para desplazarse hasta el final, pulse Inicio. Para desplazarse hacia abajo, pulse Fin.
- Para retroceder, pulse Alt + flecha izquierda. Para avanzar, pulse Alt + flecha derecha.
- Para ir al siguiente marco o barra de herramientas, pulse Ctrl + Tabulador (Ctrl+F6 si utiliza Mozilla o un navegador basado en Mozilla).
- Para desplazarse al marco anterior, pulse Mayús + Ctrl + Tabulador. (Shift+Ctrl+F6 si se utiliza Mozilla o un navegador basado en Mozilla).
- Para pasar al marco que muestra el contenido del tema, pulse Alt + K (cuando utilice el navegador de ayuda incorporado en Windows o Internet Explorer).
- Para desplazarse a la pestaña Contenido, pulse Alt + C.
- Para pasar a la pestaña Resultados de la búsqueda, pulse Alt + R.
- Para moverse entre pestañas, pulse las flechas Derecha/Izquierda.
- Para cambiar a otra vista, seleccione una pestaña y, a continuación, pulse Intro.
- Para cambiar y desplazarse a una vista, seleccione una pestaña y, a continuación, pulse la flecha arriba.
- Para desplazarse al campo de entrada de búsqueda, pulse Alt + S.
- Para imprimir la página actual o el marco activo, pulse Ctrl + P.
- Para buscar una serie en la página actual o el marco activo, pulse Ctrl + F (cuando utilice el navegador de ayuda incorporado en Windows o Internet Explorer).

La mayoría de las etiquetas de los controles en los diálogos emergentes del sistema de ayuda tienen nemotécnicos asignados. Para acceder al control asociado a una etiqueta, utilice la tecla Alt junto con la letra subrayada.

Preparación para depurar

Para poder iniciar una sesión de depuración, el daemon de la interfaz de usuario del depurador (daemon de depuración) debe estar a la escucha del motor del depurador de lenguajes compilado (motor de depuración). Además, la aplicación debe compilarse con las opciones de depuración adecuadas.

Acerca de esta tarea

Para obtener información sobre cómo establecer el daemon de depuración para que escuche los motores de depuración, consulte los temas relacionados.

Para depurar el programa en el nivel de código fuente, debe compilar el programa con determinadas opciones de compilador que indican al compilador que genere información simbólica y enganches de depuración en el archivo de objeto. Compile sin optimización (NOOPTIMIZE) y con la opción -g o TEST .

Opción -g
OPTIMIZE, opción
Utilice OPTIMIZE para reducir el tiempo de ejecución del programa objeto. La optimización también puede reducir la cantidad de almacenamiento que utiliza el programa objeto. Las optimizaciones realizadas incluyen la propagación de constantes, la planificación de instrucciones y la eliminación de cálculos cuyos resultados nunca se utilizan.

opción TEST
Utilice TEST para generar código de objeto que contenga información de símbolos y sentencias que permita al depurador realizar una depuración simbólica a nivel de fuente.

Escucha de motores de depuración

El daemon de depuración es la parte de la interfaz de usuario que escucha una conexión de motor.

Acerca de esta tarea

Puede iniciarlo realizando una de las tareas siguientes:

- Pulse el icono de daemon ( ). El icono cambiará para indicar que el daemon se ha iniciado.
- Pulse la flecha abajo a la derecha del icono de daemon y seleccione **Empezar a escuchar en el puerto: < número de puerto >** en el menú.

Para verificar si el daemon de depuración está a la escucha de motores de depuración, existen tres formas:

- Observe el estado del icono de daemon en la vista Depurar. Si el daemon está a la escucha, el icono aparece como . Si el daemon no está a la escucha, el icono aparece como .
- Pulse la flecha abajo a la derecha del icono de daemon. Si el daemon está a la escucha, el primer elemento de menú indica **El daemon de interfaz de usuario de depuración está a la escucha en el puerto: < número de puerto >**. Si el daemon no está a la escucha, el primer elemento de menú indica **Iniciar la escucha en el puerto: < número de puerto >**.
- Pase el cursor por encima del icono de daemon. Si el daemon está a la escucha, la ayuda contextual indica **El daemon de interfaz de usuario de depuración está a la escucha en el puerto: < número de puerto >. Seleccione este botón para dejar de escuchar**. Si el daemon no está a la escucha, la ayuda contextual indica que el daemon de interfaz de usuario de depuración de **no está a la escucha. Seleccione este botón para empezar a escuchar en el puerto: < número de puerto >**.

Es posible que desee detener el daemon de depuración por motivos de seguridad o si el número de puerto del daemon es necesario para otro usuario en una máquina de varios usuarios de. Sin embargo, el daemon debe estar a la escucha para iniciar una sesión de depuración de lenguaje compilado.

Para detener el daemon de depuración cuando está a la escucha, puede realizar una de las tareas siguientes:

- Pulse el icono de daemon ( ). El icono cambiará para indicar que el daemon se ha detenido.

- Pulse la flecha abajo situada a la derecha del icono de daemon y seleccione **Detener escucha** en el menú.

El puerto predeterminado utilizado por el daemon de depuración para escuchar los motores de depuración es 8001. Puede cambiar el número de puerto del daemon desde la vista Depurar o desde la página de preferencias Daemon de depuración, y puede especificar un rango de puertos para que el daemon de depuración escuche.

Para cambiar el número de puerto desde la vista Depurar, siga estos pasos:

1. Pulse la flecha abajo situada a la derecha del icono de daemon y seleccione **Cambiar puerto** en el menú.
2. Se abre un recuadro de diálogo Preferencias. En el campo **Puerto de daemon** , especifique el número de puerto o el rango de números de puerto (que se describe más adelante en este tema) que desea utilizar.
3. Pulse **Aceptar** para cambiar el número de puerto. Para revertir el número de puerto a su valor predeterminado, puede pulsar el pulsador **Restaurar valores predeterminados** .

Para cambiar el número de puerto desde la página de preferencias del daemon de depuración, consulte el tema de preferencias de depuración relacionado.

Para especificar un rango de números de puerto, separe los valores mediante comas y guiones. Por ejemplo, si especifica `8001,8003,8900-8903` , el daemon de depuración utilizará el primer puerto que esté disponible en este rango de números: 8001, 8003, 8900, 8901, 8902 y 8903. Una vez establecida la conexión del daemon, puede pasar el cursor por encima del icono del daemon y leer (en la ayuda contextual) qué puerto se ha utilizado, o puede pulsar la flecha abajo situada a la derecha del icono del daemon, donde el número de puerto está disponible en el menú.

Nota:

- A menos que el puerto ya esté en uso en el sistema (recibirá un mensaje en el cliente si este es el caso), se recomienda que utilice el puerto predeterminado.
- Si el puerto de daemon establecido anteriormente está actualmente en uso para una sesión de depuración en el entorno de trabajo, el cambio del puerto de daemon no afectará a las conexiones anteriores creadas a través del puerto. El nuevo número de puerto se utilizará para las conexiones de motor posteriores.
- Si otra aplicación ya está utilizando el nuevo número de puerto, aparecerá un mensaje de error cuando el daemon intente escuchar en el nuevo puerto. En este caso, elija un número de puerto de daemon que no esté siendo utilizado por otra aplicación.

Soporte SSL para el daemon de depuración
Se puede utilizar un daemon de depuración seguro SSL además del daemon de interfaz de usuario de depuración tradicional. Sin embargo, SSL sólo funcionará si se utiliza con un depurador remoto que lo admita.

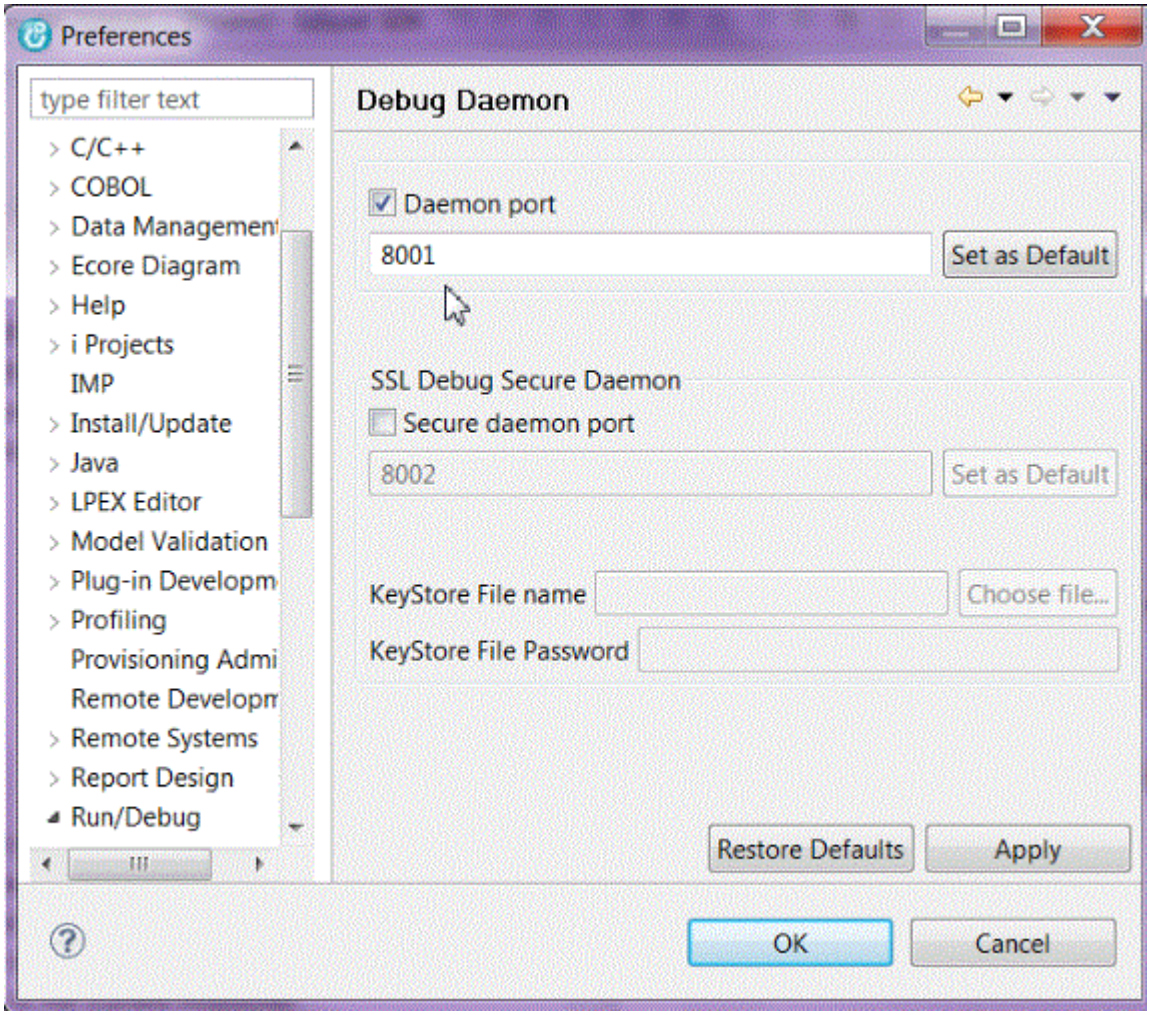
Acerca de esta tarea

Nota: Si está utilizando el túnel SSH para proteger la conexión de depuración, la conexión se debe realizar con el **puerto de daemon**.

Hay dos formas de habilitar el daemon de depuración segura SSL:

- Pulse la flecha abajo situada a la derecha del icono de daemon  y seleccione **Cambiar puerto ...**
- Abra la página de preferencias pulsando **Ventana > Preferencias**. Expanda **Ejecutar/Depurar** en el menú y seleccione **Depurar daemon**.

Aparecerá el siguiente diálogo:



Para habilitar el daemon seguro SSL, seleccione el recuadro de selección Depurar daemon seguro SSL y especifique el puerto. También se deben definir un archivo de almacén de claves y una contraseña.

Para especificar un rango de números de puerto, separe los valores mediante comas y guiones. Por ejemplo, especificar 8001.8003.8900-8903 hará que el daemon de depuración utilice el primer puerto que está disponible en este rango de números: 8001, 8003, 8900, 8901, 8902 y 8903.

Obtención de la dirección IP para la máquina cliente desde la interfaz de usuario del depurador
Para poder iniciar una sesión de depuración, se necesita la dirección IP de la máquina que ejecuta el cliente depurador (interfaz de usuario).

Acerca de esta tarea

Para obtener la dirección IP de la máquina cliente que ejecuta la interfaz de usuario del depurador, siga estos pasos:

Procedimiento

1. En la vista Depurar, pulse la flecha abajo situada a la derecha del icono de daemon y seleccione **Obtener IP de estación de trabajo** en el menú.
2. Se abrirá el mensaje Obtener IP de estación de trabajo, que indica la dirección IP actual de la máquina cliente.

Resultados

Puede seleccionar la dirección IP en este cuadro de diálogo y copiarla y pegarla.

Nota: Si la estación de trabajo tiene varios adaptadores LAN, o si hay un direccionador o una red privada virtual (VPN) entre la estación de trabajo y el servidor, este diálogo puede listar más de una dirección IP. Puede que tenga que probar cada dirección IP para encontrar la dirección que puede utilizar el servidor.

Opciones de compilador de depuración

Las opciones de compilador que son relevantes para depurar programas COBOL for Linux on x86 incluyen:

Opción de compilador	Definición
-g	Solicita al compilador que genere información de depuración para el código fuente. Debe especificar esta opción si tiene previsto depurar el código.
TEST	Equivalente a -g.

Establecimiento de preferencias de depuración

Puede establecer una variedad de preferencias relacionadas con la depuración, como por ejemplo el número de puerto del daemon que se va a utilizar, las preferencias del Editor del depurador y el periodo de tiempo que se debe esperar una respuesta del motor de depuración.

Acerca de esta tarea

Al seleccionar **Ventana** > **Preferencias** en la barra de menús del Eclipse IDE se abre el recuadro de diálogo Preferencias. En este recuadro de diálogo, puede elegir y expandir el nodo **Ejecutar/Depurar** para establecer una variedad de preferencias de depuración. Estas incluyen las preferencias siguientes (que se encuentran en los nodos **Recorrer todo animado en**, **Depurar daemon**y **Depurar compilado**) que es posible que desee establecer al depurar las aplicaciones de lenguaje compilado:

Preferencias de recorrer todo animado

Acerca de esta tarea

En el recuadro de diálogo Preferencias, al seleccionar **Ejecutar/Depurar** > **Depurar compilado** > **Recorrer todo animado** se abrirá la página de preferencias Recorrer todo animado. En esta página, puede establecer el paso actual en el ritmo (o el paso actual en el retardo) y el ritmo máximo (o el paso máximo en el retardo) del paso animado en la acción. Además, puede establecer la cantidad de tiempo durante el cual el ritmo aumenta o disminuye cuando selecciona las acciones **Acelerar Arriba** O **Bajar hacia abajo** Animadas en la vista Depurar.

Los valores predeterminados de los campos de esta página de preferencias son:

- Campo **Current pace (ms)** : 2 segundos o 2000 milisegundos
- Campo **Velocidad de subida/bajada en (ms)** : 200 milisegundos
- Campo **Peso máximo (ms)** : 5 segundos o 5000 milisegundos

Preferencias de daemon de depuración

Acerca de esta tarea

En el recuadro de diálogo Preferencias, al seleccionar **Ejecutar/Depurar** > **Daemon de depuración** se abrirá la página Daemon de depuración. En esta página, puede establecer el puerto, un rango de puertos o una combinación de puertos en los que el daemon escuchará las conexiones del motor de depuración. Los rangos y combinaciones de puertos se pueden especificar en listas separadas por comas, rangos con guiones o una combinación de los dos. De forma predeterminada, el puerto se establece en 8001.

Nota: Se recomienda que el puerto predeterminado se deje tal cual, a menos que tenga problemas o se esté ejecutando en una máquina multiusuario en la que ya se esté utilizando el puerto predeterminado.

Si cambia el puerto de daemon en la página de preferencias Daemon de depuración, puede volver a establecerlo fácilmente en su valor predeterminado pulsando el pulsador **Restaurar valores predeterminados** de la página de preferencias.

Si el daemon ya estaba establecido en la interfaz de usuario para escuchar los motores de depuración, el depurador iniciará el daemon en el nuevo número de puerto cuando cambie el número de puerto del daemon en esta página de preferencias.

Preferencias del editor del depurador

Acerca de esta tarea

En el recuadro de diálogo Preferencias, al seleccionar **Ejecutar/Depurar** > **Depurar compilado** > **Editores de depuración** se abrirá la página Editor del depurador. En esta página, puede establecer el editor para permitir la evaluación de tipo y ayuda contextual. Cuando el recuadro de selección **Permitir evaluación contextual** está seleccionado, puede pasar el cursor por encima de una expresión en el Editor del depurador para visualizar su valor en una ventana emergente. Cuando el recuadro de selección **Visualizar tipos al pasar el cursor por encima** está seleccionado, el tipo de expresión se mostrará en la ventana emergente.

El recuadro de selección **Utilizar siempre al depurar** determina el editor en el que se abrirá el origen al depurar. También determina lo que verá al pisar. El valor predeterminado para este recuadro de selección depende del producto con el que haya instalado este depurador. Cuando se deselecciona este recuadro de selección:

- El origen se abrirá en el editor predeterminado que está asociado con el tipo de archivo de origen en las preferencias del entorno de trabajo.
- Si el motor de depuración de host sólo puede encontrar el origen o el listado, se abrirá en el editor del depurador.

En esta sección, también puede:

- Establezca el editor para cargar archivos de origen completos. De forma predeterminada, este valor está desactivado. Cuando el recuadro de selección **Cargar todo el contenido del archivo** está seleccionado, se cargará todo el archivo de origen, sin embargo, el rendimiento puede verse afectado negativamente. Es posible que desee activar este valor cuando utilice determinadas acciones avanzadas del editor LPEX, como por ejemplo buscar de forma incremental en el archivo o utilizar funciones de coincidencia de corchetes.
- Establezca el depurador para permitir que las expresiones supervisadas se añadan a la vista Supervisores cuando se pulsan dos veces en el editor.
- Seleccione el recuadro de selección **Centrar vista en línea de ejecución** si desea que la línea de ejecución actual esté centrada en el editor del depurador para todas las sesiones de depuración.
- Elija el color de la línea de ejecución.

Preferencias de depuración compilada

Acerca de esta tarea

En el recuadro de diálogo Preferencias, al seleccionar **Ejecutar/Depurar** > **Depurar compilado** se abrirá la página Depurar compilado. En esta página, puede establecer estas preferencias:

Perfiles de programa

Acerca de esta tarea

Puede optar por suprimir perfiles de programa. El depurador guarda un perfil de programa para cada programa que depure. El perfil de programa incluye información como, por ejemplo, los valores de punto de interrupción y supervisor. Para suprimir todos los perfiles de programa guardados actualmente, seleccione este botón.

Si desea que los valores de punto de interrupción de excepción se apliquen sólo al programa que se está depurando en la sesión de depuración actual, seleccione el recuadro de selección **Guardar valores de punto de interrupción de excepción por programa** . Si este recuadro de selección no está seleccionado, los valores de punto de interrupción de excepción se aplicarán a todos los programas depurados por el motor de depuración actual.

Tiempo de respuesta del motor

Acerca de esta tarea

Si desea especificar el periodo de tiempo que el depurador debe esperar una respuesta del motor de depuración, seleccione el botón de selección **Esperar (en segundos)** y, a continuación, especifique el periodo de tiempo en segundos que se debe esperar en el campo. De forma predeterminada, el depurador esperará 15 segundos para obtener una respuesta del motor. Cuando el botón de selección **Esperar** está seleccionado, si un motor no responde dentro del periodo de espera especificado, un recuadro de diálogo le solicitará que continúe esperando una respuesta del motor. Si elige no continuar esperando, la sesión de depuración terminará.

Si desea que el depurador espere indefinidamente una respuesta del motor de depuración, seleccione el botón de selección **Infinito** . Cuando se selecciona este botón de selección, tendrá que terminar manualmente la sesión de depuración si un motor no puede responder.

El valor **Rastrear conexión de motor** se utiliza para fines de diagnóstico. Cuando se selecciona este valor, los archivos grandes que solo son legibles por IBM se pueden escribir en el disco. Seleccione este valor sólo cuando se lo indique un representante de servicios IBM .

Motor de depurador para lenguajes compilados

Con el diseño de cliente/servidor del depurador, puede depurar programas que se ejecutan de forma remota en otros sistemas de una red, utilizando los recursos locales de la estación de trabajo para presentar y controlar la sesión de depuración.

El programa de fondo del depurador, también conocido como *motor de depuración*, se ejecuta en el mismo sistema que el programa que desea depurar. Este sistema puede ser cualquier sistema Linux en x86 accesible a través de una red.

Nota: El motor de depuración que se suministra con este producto se identifica a sí mismo como un motor de la versión 1.0 .

Inicio del motor del depurador

Al depurar desde el cliente de interfaz de usuario, se inicia el motor del depurador utilizando la *modalidad de daemon de interfaz de usuario*. En esta modalidad, la interfaz de usuario se inicia primero y espera a que el motor se conecte a ella.

El mandato `irmtdbgc` inicia el motor de depuración en el sistema remoto. El mandato `irmtdbgc` tiene la sintaxis, `irmtdbgc [debugger parms] debuggee_name [debuggee parms]`, donde `[debugger parms]` son, en cualquier orden:

Parámetro	Descripción
`-qhost= <host:port>`	`<host>` especifica el nombre de host de la máquina que ejecuta la interfaz de usuario del depurador. Puede ser un nombre de host o una dirección IP. Si no se especifica, se utiliza el valor de la variable de entorno DER_DBG_ADDR. Si no se especifica ninguno, se utiliza el valor `localhost` . `<port>` es opcional (de forma predeterminada, se presupone el puerto 8001).

Parámetro	Descripción
`-i`	Si está presente, especifica que el depurador se detendrá inmediatamente después de cargar el depurador y no se ejecutará en el punto de entrada principal de la aplicación.
`-a xxxx`	`xxxx` puede ser un identificador de proceso o, si el nombre de la aplicación es exclusivo, el nombre del proceso tal como se muestra en el mandato `ps` .
`-qconsole=<remote, local, or GUI>`	Esto controla dónde aparecerá la consola para el programa que se está depurando. Si se especifica `-qconsole=remote` , la salida se dirigirá a la sesión local y a la interfaz de usuario. Si se especifica `-qconsole=local` , la consola aparece en la ventana de consola en la que ha escrito el mandato `irmtdbgc` . Si se especifica `-qconsole=GUI` , la consola aparece en una ventana aparte. El valor predeterminado de este parámetro es `remote`.
`-s`	Especifica que el depurador se va a ejecutar inmediatamente. El depurador se detendrá cuando alcance un punto de interrupción del perfil, o si se produce una señal.
`--`	Esto indica que el siguiente parámetro es el nombre de debuggee. Sólo es necesario si el nombre de debuggee empieza con el carácter '-'.

El depurador buscará el programa para depurar utilizando la variable de entorno *PATH* .

Variables de entorno para el motor del depurador

Las variables de entorno del motor de depuración se establecen en el entorno Linux .

Las variables de entorno siguientes controlan el comportamiento del motor:

Variable de entorno	Descripción
VÍA_ACCESO_BD_DERT	Especifica un conjunto de vías de acceso que el depurador debe utilizar para buscar archivos fuente. Estas vías de acceso se utilizarán si la información de depuración no contiene nombres de archivo de origen completos.

Variable de entorno	Descripción
DER_DBG_ADDR	Especifica el host predeterminado que se utilizará en modalidad de daemon de interfaz de usuario. Puede ser un nombre de host o una dirección IP. El valor predeterminado es `localhost`. Esto se altera temporalmente mediante el parámetro de línea de mandatos `-qhost`. Al especificar la dirección, también puede incluir el puerto predeterminado que se utilizará en la modalidad de daemon de la interfaz de usuario. Para incluir un número de puerto, especifique DER_DBG_ADDR=<host name or address>:<port>. De forma predeterminada, el número de puerto utilizado es 8001. Cualquier puerto especificado con esta variable de entorno se altera temporalmente mediante el parámetro de línea de mandatos `-quiport`.
DER_DBG_TRACE	Utilice esta variable de entorno para especificar la ubicación del archivo de rastreo del motor.
DER_DBG_PICLDUMP	Utilice esta variable de entorno para especificar la ubicación del archivo de rastreo EPDC.

Consideraciones sobre cortafuegos

Si hay un cortafuegos entre el motor y la interfaz de usuario, deberá proporcionar las reglas de cortafuegos adecuadas para que pueda tener lugar la comunicación entre el motor y la interfaz de usuario.

El motor debe estar dirigido para conectarse a la dirección IP de WAN del cortafuegos en un puerto que el cortafuegos ha configurado para reenviar al cliente. Por ejemplo:

```
Client - ip 10.1.1.7, daemon port 8101
Firewall - wan ip 10.10.10.3, configured to forward to port 8101 to 10.1.1.7
Server - start the engine to connect to the client:
irmtdbgc -qhost=10.10.10.3:8101 a.out
```

Notas:

- Muchos cortafuegos bloquean el puerto 8001. Es posible que tenga que utilizar un puerto diferente, como en el ejemplo anterior de .
- También puede dirigir el motor de depuración para conectarse a las estaciones de trabajo a través de un túnel de shell seguro (ssh), consulte la nota técnica 1438892, Depuración a través de un túnel de Secure Shell. Si está iniciando la sesión de depuración a través de una configuración de depuración en el cliente de la estación de trabajo, puede haber una opción para realizar un túnel en las conexiones en el separador **Avanzado** de la configuración de depuración.

Depuración de las aplicaciones

Después de iniciar una sesión de depuración, hay vistas de depuración disponibles que habilitan una variedad de tareas de depuración.

Acerca de esta tarea

Las vistas que están disponibles para la depuración incluyen:

- La vista Depurar: gestionar depuración de programa.
- El editor del depurador: muestra el fuente de la aplicación.

- Vista Puntos de interrupción: establecer y trabajar con puntos de interrupción.
- La vista Variables: lista y edita variables en la aplicación. También puede encontrar más información en “Inspección de variables” en la página 352.
- La vista Registros: visualizar registros en el programa.
- La vista Supervisores: trabajar con variables, expresiones y registros que elija supervisar.
- La vista Módulos: muestra una lista de los módulos cargados al ejecutar el programa. Puede navegar a las unidades de compilación individuales y a los archivos fuente de la aplicación, ver puntos de entrada de función y establecer puntos de interrupción en ellos.
- La vista Consola: muestra la salida de pantalla del programa.
- La vista Memoria: ver y correlacionar la memoria utilizada por la aplicación.

Depurador de lenguajes compilados

El depurador de lenguajes compilados es un depurador de nivel de fuente interactivo. Funciona en un cliente que está conectado a través de una conexión de red con el motor de depuración. El depurador de lenguajes compilados le permite depurar programas COBOL.

El depurador muestra los archivos fuente de la aplicación y las funciones de dichos archivos fuente. Puede realizar un solo paso, recorrer paso a paso, recorrer paso a paso o detener la ejecución en una línea o condición especificada. Al controlar la ejecución, puede supervisar variables, registros, memoria, pilas de llamadas y otros elementos.

El depurador de lenguajes compilados está habilitado para Internet Protocol Versión 6 (IPv6).

El editor del depurador

Cuando inicia una sesión de depuración, el depurador utiliza el Editor del depurador para visualizar el fuente. Este editor ofrece varias acciones de depuración.

Cuando se inicia una sesión de depuración, el origen se abre en el editor en modalidad de examen y no se puede modificar. El origen sólo se puede modificar en el Editor de depurador cuando se abre con el editor fuera de una sesión de depuración.

Cuando no se puede encontrar el origen, el editor se abre sin el origen. Para obtener información sobre cómo localizar el origen, consulte “Localizando origen” en la página 342.

Tareas relacionadas

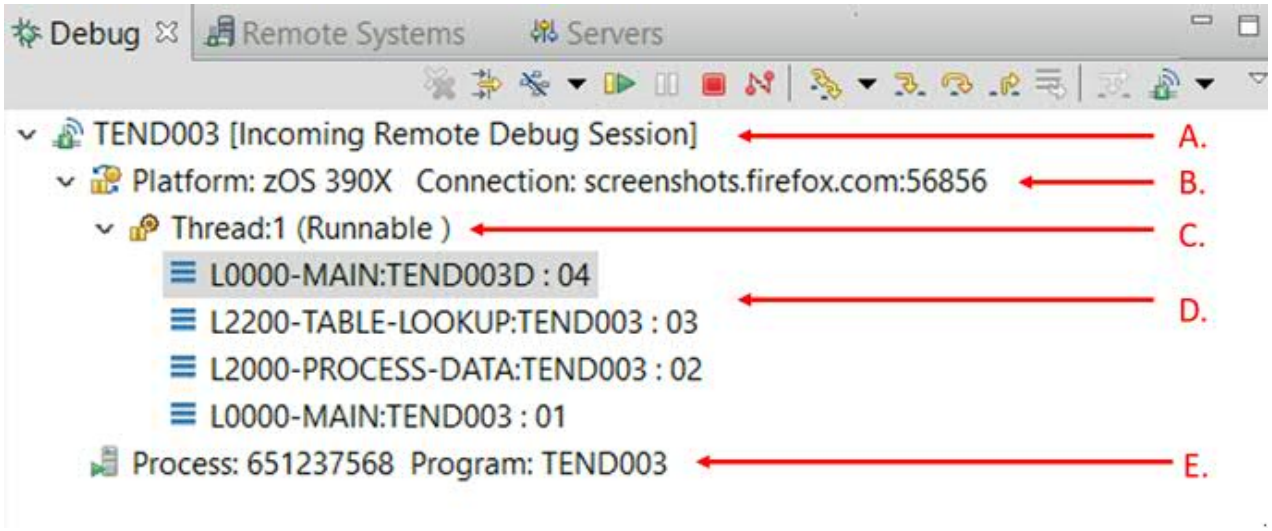
“Conmutación entre distintas vistas de depuración” en la página 344
El menú **Conmutar vista** del editor del depurador se puede utilizar para conmutar entre distintas vistas de depuración durante una sesión de depuración. Utilice las acciones **Establecer vista predeterminada** para seleccionar una vista de depuración como vista predeterminada.

Utilización de la vista Depurar

Con la vista Depurar, puede gestionar la depuración de un programa. Muestra la pila para las hebras suspendidas para cada destino que está depurando. Los destinos de depuración (asociados con hebras y marcos de pila) se visualizan en la vista Depurar para cada programa o aplicación que está depurando.

Acerca de esta tarea

En la vista Depurar, cada hebra del programa se visualiza como un nodo en el árbol. Un destino de depuración típico en la vista Depurar se describe según este diagrama:

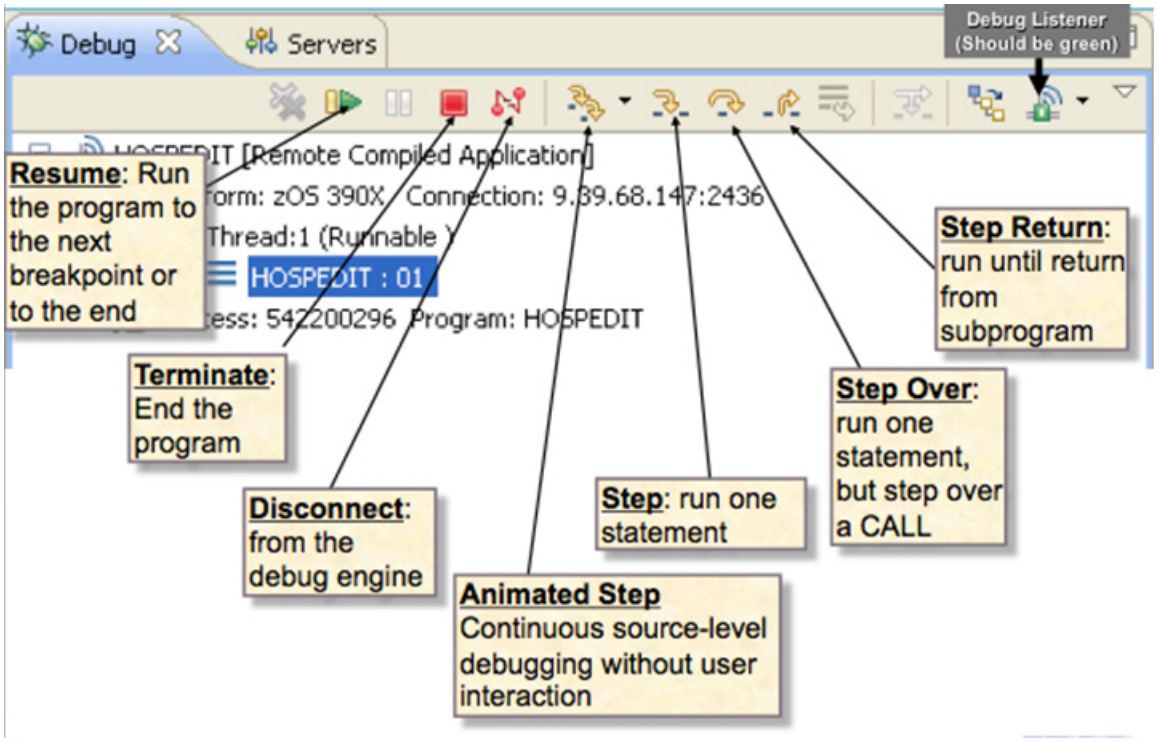


En la vista Depurar, los lanzamientos utilizados para iniciar la sesión de depuración para el programa se visualizan en el nivel de nodo superior (puntero A. en el diagrama). Debajo del lanzamiento, se muestra un nodo que representa el motor de depuración (puntero B. en el diagrama). A continuación, se muestra cada hebra del programa (puntero C. en el diagrama). Cuando la ejecución del programa se detiene, de forma predeterminada, el nodo para la hebra de detención se expande automáticamente para mostrar sus marcos de pila (puntero D. en el diagrama). Si expande manualmente otras hebras, estas hebras se expandirán automáticamente la próxima vez que se suspenda el programa. Finalmente, se muestra un nodo que representa el proceso y el programa que se está depurando (puntero E. en el diagrama).

Nota: El marco de párrafo no está soportado.

Cuando se suspende la ejecución del programa, el origen del marco de pila seleccionado se abre en el editor, resaltando la línea fuente que el programa está a punto de ejecutar. Si hay muchas hebras en el programa, la pila de la hebra que ha causado la detención se puede desplazar fuera del final del marco de depuración.

Las secciones siguientes explican las acciones que se pueden realizar utilizando los iconos de la barra de herramientas en la vista Depurar. Tal como se muestra en el diagrama siguiente, la vista Depurar también se puede utilizar para establecer el daemon de depurador. Para obtener información sobre esto, consulte el tema relacionado sobre la escucha de motores de depuración.



Ejecución, terminación y desconexión de un programa

Acerca de esta tarea

Puede realizar estas acciones de depuración básicas en la vista Depurar:

- Para ejecutar la aplicación, pulse **Reanudar** o pulse F8.
- Para terminar la sesión de depuración, pulse Terminar ( ) o pulse Shift+F8 -o pulse con el botón derecho del ratón el destino de depuración (o una de sus hebras o pilas) que desea terminar y elija una de las acciones de terminación.
- Para desconectarse del programa y dejarlo en ejecución, pulse Desconectar ( ). Esta acción puede no estar disponible, en función de cómo se haya iniciado el programa que está depurando.

Recorrer un programa

Acerca de esta tarea

Cuando se suspende una hebra, los controles de paso se pueden utilizar para recorrer la ejecución del programa línea por línea. Al realizar una operación de paso, si se encuentra un punto de interrupción o suceso, la ejecución se suspende en el punto de interrupción o suceso y la operación de paso finaliza. Puede utilizar **mandatos de paso** para recorrer el programa una única instrucción o ubicación a la vez.

Están disponibles los siguientes mandatos de paso:

- **Recorrer** ( ) (F6): al emitir un recorrer, el programa pasa a la siguiente línea de origen.
- **Recorrer** ( ) (F5): Al emitir un paso en, el programa pasará a la siguiente sentencia. Si la línea actual contiene una llamada a otra función, el depurador se detendrá en esa función.

  El comportamiento de este mandato se ve afectado por la acción **Utilizar filtros de paso** ( ) (Shift+F5). Si el filtro está desactivado (pulsador no seleccionado), el depurador se detendrá en una función llamada incluso si no contiene información de depuración y se debe visualizar el desensamblado. Si el filtro está activado (pulsador seleccionado), el depurador sólo se detendrá en

la función llamada si se puede visualizar el fuente. Si el origen no se puede visualizar, se comporta como si hubiera emitido un **Saltar**. La variable de entorno del motor de depuración *DER_DBG_ STEP_DEBUG* afecta al comportamiento de la acción **Utilizar filtros de paso** .

Nota: Para COBOL, el paso a la acción normalmente se comportará como si la acción de filtro de paso estuviera siempre activada. Al depurar programas escritos en estos lenguajes, el depurador intentará detenerse en el código fuente.

- **Retorno de paso** () (F7): cuando emite un retorno de paso, el programa se ejecuta en el punto del programa de llamada inmediatamente después de la llamada a la función actual. Normalmente se detendrá en la ubicación que sigue a la instrucción de llamada. Si el programa de llamada tiene información de depuración, puede estar en medio de una línea fuente.
- **Recorrer todo animado**(): cuando emite esta acción, el depurador emite un recorrer todo lo que se ha hecho repetidamente. Puede controlar el retardo entre cada paso seleccionando de nuevo la acción **Recorrer todo animado** .

Tareas relacionadas

“Utilización de puntos de interrupción” en la página 344
Los puntos de interrupción son marcadores temporales que se colocan en el programa ejecutable para indicar al depurador que detenga el programa en un punto determinado. Cuando se encuentra un punto de interrupción, la ejecución se suspende en el punto de interrupción antes de que se ejecute la línea, momento en el que puede ver la pila para la hebra y comprobar el contenido de las variables, registros y memoria. A continuación, puede pasar por encima (ejecutar) la línea y ver qué efecto tiene en el argumento.

Localizando origen

Cuando depura una aplicación, el motor de depuración intenta encontrar el origen de la aplicación. Si el motor de depuración puede localizar el origen, abre el origen en el editor del depurador. Si el motor de depuración no puede localizar el origen, abre una vista Desensamblado del origen en el editor del depurador.

Acerca de esta tarea

Puede utilizar este método para ayudar al motor de depuración a localizar archivos de origen:

- En la vista Depurar o en el editor del depurador, puede añadir una ubicación de origen. Por ejemplo, **Editar búsqueda de origen** abre el diálogo Editar vía de acceso de búsqueda de origen en el que puede seleccionar el tipo de ubicación de origen a añadir. De forma alternativa, puede modificar la lista de ubicaciones de origen pulsando con el botón derecho del ratón en un marco de pila o hebra de la vista Depurar y seleccionando la acción **Editar búsqueda de origen** .

Modificación de la lista de ubicaciones de origen

Acerca de esta tarea

Después de iniciar una sesión de depuración, puede modificar o añadir a la lista de ubicaciones de origen realizando estos pasos:

Procedimiento

1. Pulse con el botón derecho del ratón el destino de depuración (o una de sus hebras o marcos de pila) y elija **Editar búsqueda de origen**.

   Se abre la ventana **Editar vía de acceso de búsqueda de origen** .
2. Realice uno de estos pasos:
   - Para añadir una ubicación de origen, pulse **Añadir**. Se abre el diálogo **Añadir origen** . Elija una de estas opciones:

- **Directorio del sistema de archivos** añade un directorio del sistema de archivos local a la lista de ubicaciones de origen. Para buscar también subdirectorios, seleccione **Buscar subcarpetas**.
- **Motor de depuración** añade el motor de depuración a la lista de ubicaciones de origen.
- **Vía de acceso del motor de depuración** añade la vía de acceso especificada en el motor del depurador a la lista de ubicaciones de origen. Si especifica varias vías de acceso, sepárelas con el separador adecuado para la plataforma del motor. Por ejemplo, utilice dos puntos (:) para motores z/OS o Linux . Los cambios en el valor **Vía de acceso del motor de depuración** entran en vigor en las sesiones de depuración posteriores.

- Para eliminar una entrada, seleccione una ubicación de origen y pulse **Eliminar**.
- Para cambiar el orden de las entradas, seleccione una ubicación de origen y pulse **Arriba** o **Abajo**.

3. Para buscar todas las instancias del nombre de archivo de origen en la lista de ubicaciones de origen, seleccione **Buscar archivos de origen duplicados en la vía de acceso**. Si el depurador encuentra varias instancias del nombre de archivo, se le solicitará que elija el archivo fuente correcto.
4. Para guardar los cambios, pulse **Aceptar**.

Cambio del archivo de origen en el editor

Acerca de esta tarea

Si se cumple alguna de las condiciones siguientes, el depurador puede localizar el origen incorrecto para el marco de pila actual:

- El origen se ha movido.
- Está depurando en un sistema distinto al sistema en el que se ha creado el programa.

Si se produce esta situación, puede cambiar el archivo de texto que se abre en el editor:

Procedimiento

1. En el editor, pulse con el botón derecho del ratón y seleccione **Cambiar archivo de texto**.
2. Especifique o busque la vía de acceso y el nombre del archivo que desea abrir.

   Nota: Si está especificando un archivo en la estación de trabajo local, especifique la vía de acceso completa y el nombre de archivo.
3. Para cargar el archivo de origen especificado en el editor y cerrar la ventana, pulse **Aceptar**.

Localización del archivo fuente en el editor

Acerca de esta tarea

Cuando no se puede encontrar el origen, el editor se abre sin el origen.

Para localizar el origen, realice uno de estos pasos:

- Para especificar un nombre de archivo de origen de editor diferente, pulse **Cambiar archivo de texto**. Examine o especifique la vía de acceso y el nombre del archivo que desea abrir.

  Nota: Para especificar un archivo en la estación de trabajo, escriba la vía de acceso completa y el nombre de archivo. La capacidad de cambiar el archivo de origen del editor depende del lenguaje, el entorno y la plataforma en los que está depurando.
- Para editar la vía de acceso de búsqueda de origen, seleccione **Añadir ubicación de origen**. Se abre la ventana **Editar vía de acceso de búsqueda de origen** . Para obtener instrucciones sobre cómo añadir una ubicación de origen, consulte “Modificación de la lista de ubicaciones de origen” en la página 342.

Para abrir una vista de desensamblado del origen, pulse **Mostrar desensamblado**.

Conmutación entre distintas vistas de depuración

El menú **Conmutar vista** del editor del depurador se puede utilizar para conmutar entre distintas vistas de depuración durante una sesión de depuración. Utilice las acciones **Establecer vista predeterminada** para seleccionar una vista de depuración como vista predeterminada.

Acerca de esta tarea

Se pueden seleccionar tres vistas en el menú **Conmutar vista** : vista Fuente expandida, vista Mixta y vista Desensamblado. La vista Fuente expandida sustituye las sentencias COPY en el origen COBOL por el contenido real del libro de copias al que hacen referencia. La vista Mixto muestra el origen expandido junto con las instrucciones de desensamblado. La vista Desensamblado muestra las instrucciones de desensamblado.

Cambio a una vista de depuración diferente

Acerca de esta tarea

Utilice las acciones Conmutar vista para conmutar a una vista de depuración diferente. El valor de la vista de depuración sólo se aplica al archivo actual en la sesión de depuración actual.

Procedimiento

1. Pulse con el botón derecho del ratón en el editor del depurador.
2. Expanda el menú **Cambiar vista** .
3. Seleccione una de las acciones Mostrar para conmutar a una vista de depuración diferente.
   Las acciones Mostrar incluyen Mostrar origen expandido, Mostrar mixto y Mostrar desensamblado.

Selección de una vista de depuración como vista predeterminada

Acerca de esta tarea

Las acciones **Establecer vista predeterminada** establecen la vista de depuración seleccionada como vista predeterminada. Conmuta el archivo depurado actualmente en la sesión de depuración actual a la vista seleccionada. Las sesiones de depuración posteriores utilizarán la vista seleccionada como vista predeterminada. Si la vista predeterminada no está disponible para un idioma o una aplicación, se selecciona la siguiente vista.

Procedimiento

1. Pulse con el botón derecho del ratón en el editor del depurador.
2. Seleccione **Cambiar vista** > **Establecer vista predeterminada**.
3. Seleccione una de las vistas de depuración como vista predeterminada.
   Las vistas de depuración incluyen Expanded Source, Mixed y Disassembly.

Utilización de puntos de interrupción

Los puntos de interrupción son marcadores temporales que se colocan en el programa ejecutable para indicar al depurador que detenga el programa en un punto determinado. Cuando se encuentra un punto de interrupción, la ejecución se suspende en el punto de interrupción antes de que se ejecute la línea, momento en el que puede ver la pila para la hebra y comprobar el contenido de las variables, registros y memoria. A continuación, puede pasar por encima (ejecutar) la línea y ver qué efecto tiene en el argumento.

Acerca de esta tarea

El depurador da soporte a los siguientes tipos de puntos de interrupción:

- **Puntos de interrupción de línea** se desencadenan cuando la línea en la que están establecidos está a punto de ejecutarse.
- Los **Puntos de interrupción de entrada** se desencadenan cuando se especifican los puntos de entrada a los que se aplican.
- Los **puntos de interrupción de dirección** se desencadenan antes de que se ejecute la instrucción de desensamblado en una dirección determinada.
- Los **puntos de interrupción de carga** se desencadenan cuando se carga una DLL o un módulo de objeto.
- Los **puntos de interrupción condicionales** se desencadenan mediante parámetros que controlan el comportamiento de estos puntos de interrupción. No todos los tipos de punto de interrupción admiten condiciones.
- Los **puntos de interrupción de suceso** se desencadenan cuando el depurador reconoce una excepción emitida por la aplicación.
- Los **puntos de interrupción de observación** se desencadenan cuando la ejecución cambia datos en una dirección específica.
- Los **puntos de interrupción de aparición** se desencadenan cuando se produce un suceso o se genera una excepción específica. Cuando se desencadena el punto de interrupción, se puede realizar una acción (opcional).

Los puntos de interrupción de suceso se establecen en la vista Puntos de interrupción pulsando el pulsador **Gestionar puntos de interrupción de suceso de lenguaje compilado** y, a continuación, en el diálogo Gestionar puntos de interrupción de suceso, seleccionando el tipo de suceso que desea que capte el depurador. Estos puntos de interrupción incluyen todas las señales estándar y una serie de sucesos de interés, como excepciones C++, y llamadas a funciones de biblioteca como `exit()`. Para las señales POSIX , puede elegir que se le notifique de todas las apariciones de cada señal individual (señales manejadas), o solo de aquellas apariciones cuando no se ha proporcionado ningún manejador (señales no manejadas).

Los puntos de interrupción de línea se pueden establecer en el editor efectuando una doble pulsación en el área de regla a la izquierda de una línea ejecutable o pulsando con el botón derecho del ratón en una acción de menú emergente en el editor al depurar -o se pueden establecer mediante el asistente en la vista Puntos de interrupción . Si desea un punto de interrupción de línea específico de hebra, debe establecerlo desde la vista Puntos de interrupción mientras haya una sesión de depuración activa. Los puntos de interrupción de entrada se pueden establecer en la vista Módulos pulsando con el botón derecho del ratón en un punto de entrada y seleccionando **Establecer punto de interrupción de entrada** en el menú emergente, o se pueden establecer mediante el asistente en la vista Puntos de interrupción. Además, puede pulsar con el botón derecho del ratón en el destino de depuración (o una de sus hebras o marcos de pila) en la vista Depurar y seleccionar **Opciones > Detener en todas las entradas de función** en el menú emergente para detenerse en todos los puntos de entrada (esta opción también está disponible en el menú emergente de la vista Puntos de interrupción). Todos los demás tipos de punto de interrupción se establecen mediante el asistente en la vista Puntos de interrupción. Para acceder a los asistentes para establecer puntos de interrupción, pulse con el botón derecho del ratón en la vista Puntos de interrupción y seleccione **Añadir punto de interrupción** en el menú emergente. Esto se expandirá a un menú que le permite elegir el tipo de punto de interrupción que desea establecer. Cuando utilice el asistente para establecer un punto de interrupción, puede especificar parámetros de punto de interrupción opcionales y establecer puntos de interrupción condicionales (consulte el tema relacionado).

La vista Puntos de interrupción muestra una lista de todos los puntos de interrupción para todas las sesiones de depuración. Puede reducir el número de puntos de interrupción visualizados de una de las maneras siguientes:

- Para filtrar puntos de interrupción que no están relacionados con la sesión de depuración actual, pulse el pulsador **Mostrar puntos de interrupción soportados por destino seleccionado** de la vista Puntos de interrupción.

- Para enlazar la vista Puntos de interrupción con la vista Depurar, pulse el conmutador **Enlazar con vista Depurar** . Cuando se selecciona este conmutador y un punto de interrupción suspende una sesión de depuración, dicho punto de interrupción se seleccionará automáticamente en la vista Puntos de interrupción.

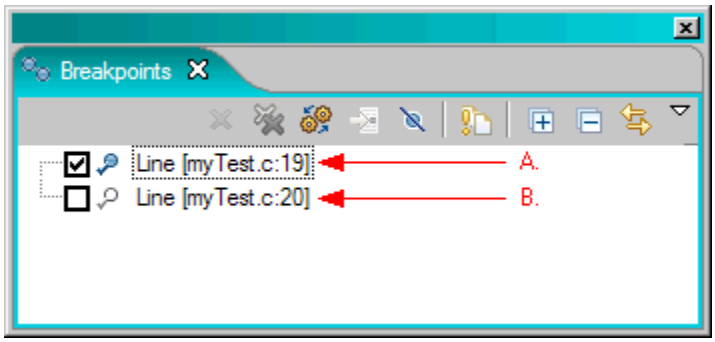
También puede agrupar puntos de interrupción para facilitar la visualización en la vista Puntos de interrupción. Los puntos de interrupción se pueden agrupar por puntos de interrupción (la lista estándar de puntos de interrupción), tipos de punto de interrupción (por ejemplo, agrupados por puntos de interrupción de línea y de entrada) y por conjuntos de trabajo de puntos de interrupción (grupos que usted mismo define). Para agrupar puntos de interrupción, seleccione el icono de flecha abajo de la vista Puntos de interrupción y, a continuación, seleccione la agrupación que desea visualizar en la vista Puntos de interrupción. Al pulsar **Avanzado** en este menú, se abre un recuadro de diálogo que le permite crear agrupaciones anidadas. Para crear conjuntos de trabajo, elija **Conjuntos de trabajo** en el menú del icono de flecha abajo de la vista Puntos de interrupción.

Las entradas de punto de interrupción de la lista le proporcionan, entre corchetes, un resumen de las propiedades de los puntos de interrupción. Con las opciones del menú emergente, puede añadir puntos de interrupción, eliminar puntos de interrupción y habilitar o inhabilitar puntos de interrupción. También puede editar propiedades de punto de interrupción con una opción de menú emergente. Con pulsadores en la vista Puntos de interrupción, puede eliminar puntos de interrupción.

Cuando elige editar un punto de interrupción, se abre el asistente mediante el cual se ha creado (si no ha utilizado un asistente para crear el punto de interrupción, se abre el asistente para el tipo de punto de interrupción). Mientras esté en el asistente, puede pulsar **Siguiente >** o **< Atrás** para ver o editar los valores de punto de interrupción en el asistente. Cuando haya terminado, pulse **Finalizar** para cambiar el punto de interrupción o pulse **Cancelar** para salir del asistente sin realizar ningún cambio.

Los puntos de interrupción se pueden habilitar e inhabilitar con menús emergentes en la vista Puntos de interrupción o el editor y mediante el recuadro de selección en la vista Puntos de interrupción. Cuando se habilita un punto de interrupción, hará que todas las hebras se suspendan cuando se llegue a él. Cuando un punto de interrupción está inhabilitado, no hará que se suspendan las hebras. Para obtener información sobre la habilitación e inhabilitación de puntos de interrupción, consulte el tema relacionado.

En la vista Puntos de interrupción, hay dos indicadores a la izquierda de un punto de interrupción establecido ( ). En el extremo izquierdo hay un recuadro de selección que indica si el punto de interrupción está habilitado. Cuando está habilitado, el recuadro de selección contiene una marca de selección. (Consulte el puntero A. en el diagrama siguiente.) Cuando está inhabilitado, el recuadro de selección no contiene una marca de selección. (Consulte el puntero B. en el diagrama siguiente.)



Un indicador con una superposición de marca de selección muestra un punto de interrupción que el motor de depuración ha instalado correctamente. Si el punto de interrupción está habilitado, este indicador se rellena; si el punto de interrupción está inhabilitado, este indicador no se rellena. En el editor, los puntos de interrupción de línea se indican mediante un indicador con un preformato de marca de selección, que indica un punto de interrupción que el motor de depuración ha instalado correctamente (si el punto de interrupción está habilitado, este indicador se rellena; si el punto de interrupción está inhabilitado, este indicador no se rellena).

Los puntos de interrupción deben estar instalados para suspender la ejecución. Es posible añadir un punto de interrupción que no es válido para la sesión de depuración actual. Este punto de interrupción no

se instalará hasta que forme parte de una sesión de depuración que incluya un motor de depuración que reconozca el punto de interrupción.

En el editor, la línea, la sentencia y los indicadores de punto de interrupción de entrada se muestran en la barra de marcadores a la izquierda del editor. Los indicadores de línea, sentencia , dirección, observación y puntos de interrupción de carga se muestran en la vista Puntos de interrupción.

Mientras esté en la vista Puntos de interrupción, el editor de origen se abrirá en la ubicación de un punto de interrupción si realiza una de las acciones siguientes:

- Efectúe una doble pulsación en el punto de interrupción.
- Seleccione el punto de interrupción y pulse el pulsador **Ir a archivo para punto de interrupción** .
- Pulse con el botón derecho del ratón en el punto de interrupción y seleccione **Ir a archivo** en el menú emergente.

Establecimiento de puntos de interrupción

Acerca de esta tarea

La información siguiente describe cómo establecer puntos de interrupción de línea , puntos de interrupción de entrada, puntos de interrupción de entrada de origen y otros tipos de punto de interrupción.

Establecer un punto de interrupción de línea

Acerca de esta tarea

Para establecer un punto de interrupción de línea , realice uno de los pasos siguientes:

- Al depurar, pulse con el botón derecho del ratón en la sentencia en la que desea detenerse y seleccione **Añadir punto de interrupción** > **Línea**.
- En el editor del depurador, efectúe una doble pulsación en el área de margen situada a la izquierda de la línea. Utilice este método para establecer o eliminar puntos de interrupción.
- En la vista Puntos de interrupción, pulse con el botón derecho del ratón en un área vacía y seleccione **Añadir un punto de interrupción**.

Establecer un punto de interrupción de entrada

Acerca de esta tarea

Para establecer un punto de interrupción de entrada, realice uno de los pasos siguientes:

- En la vista Módulos, pulse con el botón derecho del ratón en un punto de entrada y seleccione **Establecer punto de interrupción de entrada**.
- En la vista Depurar, pulse con el botón derecho el destino de depuración o una de sus hebras o marcos de pila y, a continuación, seleccione **Opciones** > **Detener en todas las entradas de función** para detenerse en todos los puntos de entrada.
- En la vista Esquema, pulse con el botón derecho del ratón en el nombre del punto de entrada y seleccione **Conmutar punto de interrupción de entrada**. No puede añadir información como, por ejemplo, expresiones condicionales cuando añade un punto de interrupción de entrada desde la vista Esquema.
- En la vista Puntos de interrupción, seleccione **Detener en todas las entradas de función**.
- En la vista Puntos de interrupción, seleccione **Añadir punto de interrupción** > **Entrada ...**.

Establecer un punto de interrupción event

Acerca de esta tarea

Para establecer un punto de interrupción event , realice estos pasos:

Procedimiento

1. En la vista Puntos de interrupción, pulse **Gestionar puntos de interrupción de sucesos de lenguaje compilado**.
2. En el diálogo Gestionar puntos de interrupción de Suceso , seleccione el tipo de suceso que desea que capte el depurador.

Establecer otros tipos de punto de interrupción

Acerca de esta tarea

Para establecer un punto de interrupción de entrada de origen, realice uno de los pasos siguientes:

Procedimiento

1. Pulse con el botón derecho del ratón en la vista Puntos de interrupción y seleccione **Añadir punto de interrupción** en el menú. Esto se expandirá a un menú completo de los tipos de punto de interrupción soportados.
2. Seleccione el tipo de punto de interrupción que desea establecer. Cuando utilice este asistente para establecer un punto de interrupción, puede especificar parámetros de punto de interrupción opcionales y establecer puntos de interrupción condicionales.

Exportar e importar puntos de interrupción

Exportar puntos de interrupción

Para exportar puntos de interrupción, realice los siguientes pasos:

Procedimiento

1. Pulse con el botón derecho del ratón en la vista Puntos de interrupción y seleccione **Exportar puntos de interrupción**.
2. En la ventana Exportar puntos de interrupción, seleccione los puntos de interrupción que desea exportar.
3. En el campo **Archivo de destino**, escriba la vía de acceso y el nombre de archivo.
4. Si no desea que el depurador le avise cuando se altera temporalmente un archivo existente con el mismo nombre, seleccione **Sobrescribir archivo existente sin avisar**.
5. Pulse **Finalizar** para guardar en el archivo.

   Puede importar los puntos de interrupción guardados en este archivo y enviar el archivo a otros usuarios que están depurando el mismo programa para que puedan importar los mismos puntos de interrupción.

Importar puntos de interrupción

Para importar puntos de interrupción, realice los siguientes pasos:

Procedimiento

1. Verifique que está importando los puntos de interrupción en el mismo programa desde el que los ha exportado. Si intenta importar el punto de interrupción a un programa distinto, es posible que el depurador no pueda instalar el punto de interrupción.
2. Pulse con el botón derecho del ratón en la vista Puntos de interrupción y seleccione **Importar puntos de interrupción**.

   Se abre la ventana Importar puntos de interrupción.
3. En el campo **Archivo de destino** de la ventana Importar puntos de interrupción, especifique la vía de acceso y el nombre de archivo, y especifique `bkpt` como extensión de archivos. También puede utilizar **Examinar** para navegar al archivo.

4. Si hay puntos interrupción que desea que el depurador sustituya por los puntos de interrupción del archivo, marque el recuadro de selección **Actualizar puntos de interrupción**.
5. Si desea que el depurador cree un conjunto de trabajo nuevo para estos puntos de interrupción, marque el recuadro **Crear conjuntos de trabajo de punto de interrupción**.
6. Pulse **Finalizar**.

Habilitar e inhabilitar puntos de interrupción

En lugar de suprimir un punto de interrupción, puede inhabilitarlo para que no detenga la ejecución del programa. Si un punto de interrupción está habilitado, provocará la suspensión de todas las hebras cuando se llegue a él. Si un punto de interrupción está inhabilitado, no provocará la suspensión de las hebras. Se pueden añadir, suprimir, habilitar o inhabilitar puntos de interrupción mientras se ejecuta la aplicación.

Acerca de esta tarea

Si se inhabilita un punto de interrupción, éste permanece en la vista Puntos de interrupción. Para que el programa se detenga en un punto de interrupción inhabilitado, selecciónelo y habilítelo. La ventaja de inhabilitar un punto de interrupción en lugar de suprimirlo es que no es necesario buscar la ubicación en el código fuente para establecer de nuevo el punto de interrupción. Además, un punto de interrupción inhabilitado guarda los valores adicionales en el punto de interrupción.

A la izquierda de un punto de interrupción establecido hay dos indicadores. En el extremo izquierdo hay un recuadro de selección que indica si el punto de interrupción está habilitado. Los puntos de interrupción habilitados se indican con una marca de selección en el recuadro, mientras que los puntos de interrupción inhabilitados se indican sin marca de selección en el recuadro. Si un punto de interrupción está inhabilitado, puede elegir **Habilitar** en su menú emergente en la vista Puntos de interrupción o en el editor (donde la opción de menú es **Habilitar punto de interrupción**). Si un punto de interrupción está habilitado, puede elegir **Inhabilitar** en su menú emergente.

Habilitar e inhabilitar puntos de interrupción desde la vista Puntos de interrupción
Para habilitar o inhabilitar un solo punto de interrupción desde la vista Puntos de interrupción:

Procedimiento

1. Pulse sobre la vista Puntos de interrupción para abrirla en primer plano.
2. Desplácese por la lista de puntos de interrupción hasta ver el punto de interrupción que desea habilitar o inhabilitar. Si desea habilitar o inhabilitar varios puntos de interrupción, selecciónelos mediante las teclas Mayús o Control.
3. Realice una de las acciones siguientes:
   - Para habilitar o inhabilitar un punto de interrupción, utilice el recuadro de selección situado en el extremo izquierdo del punto de interrupción. Para habilitar un punto de interrupción, marque el recuadro de selección. Para inhabilitar un punto de interrupción, quite la marca del recuadro de selección.
   - Pulse con el botón derecho del ratón en el punto de interrupción que desea habilitar o inhabilitar y seleccione **Habilitar** o **Inhabilitar**.

Habilitar e inhabilitar puntos de interrupción del editor
Para habilitar o inhabilitar un solo punto de interrupción desde el editor:

Procedimiento

1. Localice el punto de interrupción en el editor.
2. Realice una de las tareas siguientes:
   - Pulse con el botón derecho del ratón el indicador del punto de interrupción en la regla del editor y seleccione **Habilitar punto de interrupción** o **Inhabilitar punto de interrupción**.

- Pulse con el botón derecho del ratón el punto de interrupción en el editor y seleccione **Habilitar punto de interrupción** o **Inhabilitar punto de interrupción** en el menú emergente.

El indicador del punto de interrupción del editor cambiará a un punto vacío si el punto de interrupción se ha inhabilitado o a un punto relleno si el punto de interrupción se ha habilitado.

Inhabilitar todos los puntos de interrupción
Para inhabilitar todos los puntos de interrupción:

Procedimiento

1. Pulse el botón conmutador **Saltar todos los puntos de interrupción**.
   Con ello inhabilitará temporalmente todos los puntos de interrupción.
2. Para volver a habilitar todos los puntos de interrupción excepto los que ha inhabilitado específicamente con la acción **Inhabilitar punto de interrupción**, pulse de nuevo el botón de conmutador **Saltar todos los puntos de interrupción**.

Puntos de interrupción condicionales

Los parámetros de punto de interrupción opcionales se utilizan para controlar el comportamiento de los puntos de interrupción.

Nota: No todos los puntos de interrupción soportan condiciones.

Cuando establece un punto de interrupción, puede hacerlo condicional estableciendo estos parámetros en la página **Parámetros opcionales** de cualquier asistente de punto de interrupción:

Parámetro de punto de interrupción opcional	Descripción	Tipo de punto de interrupción soportado
Subproceso	Los puntos de interrupción pueden ser específicos de hebra. Puede especificar si el punto de interrupción se aplica a todas las hebras (el valor predeterminado) o sólo a una hebra específica (n=one). Para especificar todas las hebras, seleccione **Cada**. Para especificar una hebra individual, elija la hebra.	Este parámetro está soportado por todos los tipos de punto de interrupción.

Parámetro de punto de interrupción opcional	Descripción	Tipo de punto de interrupción soportado
Frecuencia	Indica cuándo se debe detener en un punto de interrupción y cuándo se debe omitir. El depurador realiza un seguimiento del número de veces que se encuentra cada punto de interrupción. Los campos de esta sección indican al depurador en qué encuentro de un punto de interrupción el depurador se detendrá primero, con qué frecuencia se detendrá y en qué encuentro el depurador ya no se detendrá. Los parámetros siguientes se utilizan para establecer la frecuencia de punto de interrupción: • **`From:`** Especifique la primera aparición de punto de interrupción en la que desea que se detenga el depurador. Por ejemplo, si desea que el depurador omita el punto de interrupción las primeras cinco veces que se encuentra, especifique 6. • **`To:`** Especifique el último punto de interrupción en el que desea que se detenga el depurador. Por ejemplo, si desea que empiece a ignorar el punto de interrupción después del encuentro 20th , especifique `20`. Para detenerse en cada encuentro, especifique `Infinity`. • **`Every:`** Especifique la frecuencia con la que desea que el depurador se detenga en este punto de interrupción. Por ejemplo, si desea que se detenga en uno de cada cuatro encuentros, especifique 4.	Este parámetro está soportado por todos los tipos de punto de interrupción.

Parámetro de punto de interrupción opcional	Descripción	Tipo de punto de interrupción soportado
Expresión	Puede especificar una expresión en este campo. La ejecución del programa se detiene en el punto de interrupción sólo si la condición especificada en este campo es verdadera (cualquier valor distinto de cero se considera verdadero). Por ejemplo, si está depurando un programa C++, puede escribir la expresión siguiente: `(i==1) || (j==k) && (k!=5)` Una expresión condicional es cualquier expresión válida en el lenguaje de la ubicación del punto de interrupción que se evalúa en un número, y no tiene efectos secundarios ni implica llamar a una función. Para C y C++, no se permiten todos los operadores de asignación y los operadores de incremento y decremento (++ y --). **Atención:** Aunque una aplicación no parece detenerse en un punto de interrupción cuya condición no se ha cumplido, el depurador suspende temporalmente la aplicación mientras evalúa la condición. Para la mayoría de los propósitos, esta pausa corta no es significativa. Sin embargo, en una aplicación multihebra, una pausa puede hacer que el sistema operativo cambie el orden en el que se asignan las hebras.	Línea, Entraday Dirección.

Inspección de variables

Acerca de esta tarea

Puede inspeccionar las variables de dos maneras: moviendo el ratón sobre el nombre de una variable en la ventana del editor de depuración (pasando el cursor por encima) o utilizando la vista Variables. Al pasar el cursor por encima, el depurador muestra el valor de una variable en una estructura de árbol en una ventana pequeña que desaparece cuando se aleja el ratón del nombre de la variable. Mientras el depurador muestra la ventana, puede expandir y contraer la estructura en árbol para ver más

información. También puede editar el valor de una variable desde la ventana contextual. La vista Variables del depurador proporciona un acceso fácil a las variables del programa y le permite observar y editar variables.

Cuando una hebra se suspende, el marco de pila superior de la hebra se selecciona automáticamente. Cuando se selecciona un marco de pila, las variables visibles en ese marco de pila se muestran en la vista Variables. Las variables complejas se pueden expandir para mostrar los elementos que componen la variable.

Procedimiento

- Los valores de variable se pueden cambiar en la ventana contextual realizando estos pasos:
  a) Seleccione una variable o un campo de la estructura de árbol en la ventana contextual.

     El valor del elemento seleccionado se muestra en el panel de detalles debajo de la estructura en árbol.
  b) Pulse dentro del panel de detalles para editar el valor de la variable.

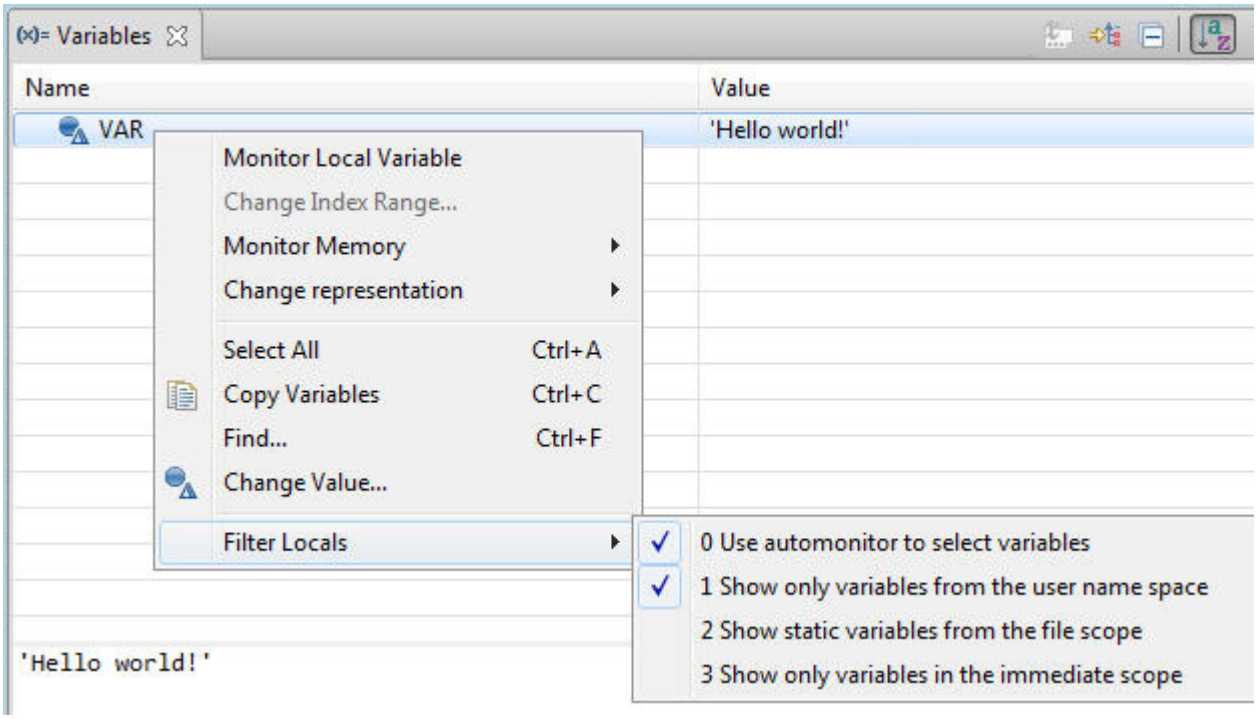
     Puede utilizar **Cortar**, **Copiar**, **Pegar**o **Seleccionar todas las acciones** cuando edite el valor en el panel de detalles.
  c) Después de editar el valor de la variable, pulse el botón **Asignar valor** debajo del panel de detalles para asignar el nuevo valor a la variable.
- Los valores de variable se pueden cambiar en la vista Variables pulsando el valor de la variable en la columna **Valor** y cambiando el valor en línea o realizando estos pasos:
  a) Pulse con el botón derecho del ratón en la variable que desea editar y seleccione **Cambiar valor** en el menú emergente.
  b) En el diálogo resultante, cambie el valor de la variable.

Resultados

Para indicar que el valor de la variable ha cambiado, su indicador tendrá un símbolo delta junto a él. Todas las variables afectadas por el cambio también tendrán un símbolo delta junto a sus indicadores.

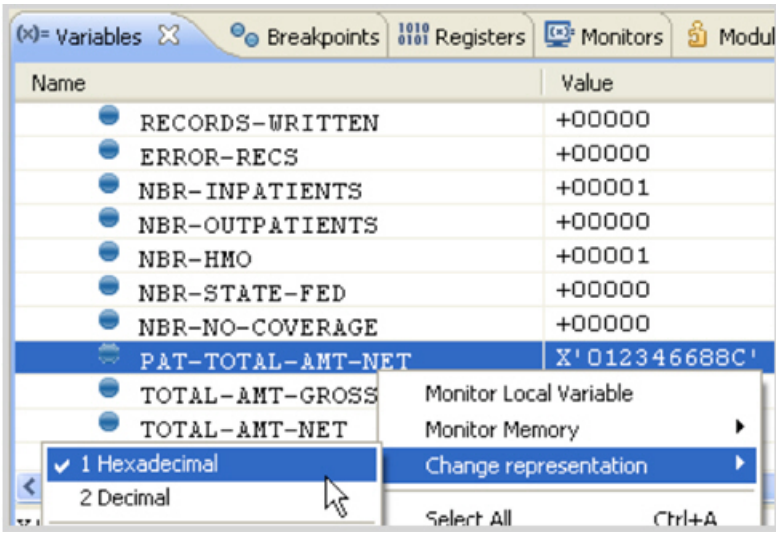
La vista Variables muestra todas las variables para un marco de pila seleccionado. La vista muestra dinámicamente las variables en el ámbito actual y aparecerán y desaparecerán a medida que se recorre o se reanuda el programa. Como alternativa a la vista Variables, puede supervisar variables en la vista Supervisores. En la vista de supervisores, el depurador siempre muestra el valor de una variable si se puede obtener. Para ver e inspeccionar una o varias variables a la vez, pulse con el botón derecho del ratón en la variable o variables y seleccione **Supervisar variable local** en el menú emergente para trabajar con las variables en la vista Supervisores.

En función del lenguaje que esté depurando, puede filtrar la vista Variables para mostrar sólo determinadas variables. Para ello, pulse con el botón derecho del ratón en la vista Variables y seleccione una entrada en el submenú **Entornos locales de filtro** tal como se muestra en la imagen siguiente.



Nota: Las opciones de filtrado y el contenido disponible en el submenú **Entornos locales de filtro** dependen del idioma que esté depurando.

En la vista de variables, puede establecer el valor para que se represente en un formato decimal o hexadecimal. Para seleccionar la representación, resalte la variable, pulse con el botón derecho del ratón, seleccione **Cambiar representación** y, a continuación, el formato deseado:



La visualización del valor en hexadecimal puede ser útil al depurar excepciones de datos.

Añadir una variable, expresión o registro a la vista Supervisores

La vista Supervisores muestra las variables, expresiones y registros que ha seleccionado supervisar. Puede especificar las variables o expresiones en un recuadro de diálogo o seleccionarlas en el editor del depurador. Utilice la vista Supervisores para supervisar variables globales o variables, expresiones y registros que desee ver en todo momento durante la sesión de depuración. Desde la vista Supervisores también puede modificar el contenido de las variables, expresiones o registros o cambiar la representación de los valores. Nota: el soporte de expresiones de algunos lenguajes de programación puede depender de la versión del compilador y/o del tiempo de ejecución de dichos lenguajes instalado en el servidor.

Acerca de esta tarea

Para añadir un Supervisor de programas nuevo para una expresión desde la vista Supervisores:

Procedimiento

1. En el editor, seleccione la línea de origen que representa el contexto en el que desea evaluar la expresión. De forma alternativa, en la vista Depurar, seleccione la hebra que contiene la expresión que desea supervisar.
2. Haga clic en el botón de la vista de monitores **Expresión de monitores** ( ).
3. En el recuadro de diálogo Supervisar expresión, especifique la variable, expresión o registro en el campo.
4. Pulse **Aceptar**.

Acciones adicionales

Acerca de esta tarea

La siguiente información describe maneras adicionales para añadir un supervisor de programa desde el editor, la vista de variables, la vista de registros, y cómo cambiar el contenido de una variable, expresión o registro en la vista de supervisores.

Para añadir un Supervisor de programas nuevo para una variable o expresión desde el editor:

1. En el editor, resalte y pulse con el botón derecho del ratón la expresión que desea supervisar.
2. Seleccione **Supervisar expresión** en el menú emergente.

Para añadir un Supervisor de programas nuevo para una variable o expresión desde la vista Variables:

1. En la vista Variables, pulse con el botón derecho del ratón la variable que desea supervisar. Para añadir varios supervisores, seleccione varias variables con las teclas Control o Mayús del teclado.
2. Seleccione **Supervisar variable local** en el menú emergente.

Para añadir automáticamente las variables de cada línea a la vista Supervisor a medida que recorre cada línea:

1. Detenga el programa en la primera línea cuya supervisión desee iniciar.
2. En la vista Consola de depuración especifique el mandato SET AUTOMONITOR ON.

Para añadir un Supervisor de programas nuevo para un registro desde la vista Registros:

1. En la vista Registros, pulse con el botón derecho del ratón el registro que desea supervisar.
2. Seleccione **Supervisar registro** en el menú emergente.

Para cambiar el contenido de una variable, expresión o registro en la vista Supervisores:

1. Seleccione la expresión cuyo valor desea modificar.
2. Si la expresión es una estructura o matriz, expándala para visualizar sus elementos individuales.
3. Desplácese a la expresión que desea cambiar y efectúe una de las siguientes acciones:
   - Efectúe una doble pulsación en la expresión.
   - Pulse la expresión con el botón derecho del ratón y elija **Cambiar valor** en el menú emergente.

   Nota: Si efectúa una doble pulsación sobre una variable y su campo de valor no puede editarse, significa que la variable es de un tipo que no puede modificarse.
4. Especifique un valor nuevo para la expresión y pulse **Intro**. El valor nuevo puede ser cualquier expresión válida que no tenga efectos colaterales. Para indicar que el valor de la expresión ha cambiado, su indicador tendrá un símbolo delta junto a ella. Todas las expresiones afectadas por el cambio tendrán también un símbolo delta junto a sus indicadores.

El depurador intentará efectuar la recuperación si una de estas restricciones no se cumple. Sin embargo, no puede garantizar que el estado de la aplicación que se depura no cambie de forma irrevocable.

Si está supervisando una variable en un programa COBOL optimizado, verá el mensaje de error siguiente siempre que ejecute una sentencia que cambie el valor de esa variable: `Se ha producido un error: EQA2421E La asignación no se ha realizado porque el programa no puede utilizar el valor asignado debido a la optimización.` El depurador no ejecuta la sentencia. Para ejecutar esa sentencia, siga estos pasos:

1. Añada la variable a la vista Supervisores mediante cualquier método descrito anteriormente. El depurador visualiza el nombre y el valor actual de la variable en la vista Supervisor.
2. Recorra el programa hasta que alcance una sentencia que altere el valor de esa variable.
3. Especifique el mandato **`SET WARNING OFF`** en la vista Consola de depuración. La vista Consola de depuración visualiza un mensaje que indica que se ha recibido el mandato **`SET WARNING OFF`**.
4. Recorra la sentencia. El valor nuevo de la variable que está supervisando se visualiza en la vista Supervisores.

Establecer la representación del contenido del supervisor

Puede cambiar la representación de las variables y expresiones en las vistas **Supervisores** y **Variables** y establecer la representación actual de una variable o expresión como valor predeterminado para la misma.

Procedimiento

1. Pulse con el botón derecho del ratón la variable o expresión para la que desea cambiar la representación.
2. Seleccione **Cambiar representación** en el menú emergente.
3. Seleccione la representación que desee. La representación actual tendrá una marca de selección junto a ella.
4. Puede establecer la representación actual como valor predeterminado del siguiente modo:
   a. Asegúrese de haber establecido la variable o expresión en la representación que desea siguiendo los pasos anteriores.
   b. Pulse en la variable o expresión con el botón derecho del ratón y seleccione **Cambiar representación > Establecer representación predeterminada** en el menú emergente.

   La representación predeterminada de la variable o expresión se utilizará en las sesiones de depuración posteriores de la aplicación.

Desreferenciar variables y expresiones

Las variables o expresiones que pueden evaluarse como una dirección pueden desreferenciarse en la vista **Variables** o en la vista **Supervisores**, si está supervisando la variable o expresión. Para desreferenciar, siga estos pasos:

Procedimiento

1. En la vista Variables o Supervisores, pulse con el botón derecho del ratón una variable o expresión que se pueda desreferenciar (por ejemplo, un puntero).
2. Elija **Desreferenciar** en el menú emergente.

Ver el contenido de un registro

Puede ver el contenido de un registro desde la vista Registros, la vista Supervisores o la vista Memoria. En estas vistas, puede observar y cambiar el contenido de los registros de la hebra actual del programa. En la vista Registros, los registros están ordenados por categorías, por lo que sólo es necesario expandir la categoría de registros que desea visualizar.

Ver el contenido de un registro en la vista Registros

Procedimiento

1. En la vista Depurar, seleccione la hebra, o el marco de pila de la hebra, cuyos registros desea ver.
2. En la vista Registros, expanda el grupo de registros que desea ver.
3. Si es necesario, utilice las barras de desplazamiento o las teclas AvPág y RePág para desplazarse por la vista Registros hasta que el registro sea visible.

   Nota: Si está en la vista Registros y desea copiar un valor, debe ponerlo en modalidad de edición para poder copiarlo. Para editar un valor en la vista Registros, pulse dos veces sobre él o púlselo con el botón derecho del ratón y seleccione **Cambiar valor** en el menú. Cualquiera de estas acciones abrirá el recuadro de diálogo Establecer valor, desde el que puede copiar el valor del registro.

   Consejo: Para mejorar el rendimiento, contraiga los grupos de registros que no esté utilizando o editando.

   Los valores de registro pueden cambiarse en la vista Registros siguiendo estos pasos:

   a. Pulse con el botón derecho del ratón el registro que desea editar y seleccione **Cambiar valor** en el menú emergente.
   b. En el diálogo resultante, cambie el valor de la variable.
   c. Pulse **Aceptar**. Para indicar que el valor del registro ha cambiado, su indicador tendrá un símbolo delta junto a él.

      Atención: Para completar el diálogo, debe pulsar **Aceptar** en lugar de la tecla Intro del teclado. Al seleccionar la tecla Intro del teclado se insertará una nueva línea en el valor del registro.

Ver el contenido de un registro que ya ha añadido a la vista Supervisores

Procedimiento

1. Si es necesario, utilice las barras de desplazamiento o las teclas AvPág y RePág para desplazarse por la vista Supervisores hasta que el registro sea visible.
2. Si desea cambiar la representación del registro, pulse el botón derecho del ratón en el nombre de registro y seleccione **Cambiar representación** en el menú emergente. A continuación, seleccione la representación que desee en la selección del menú emergente resultante.

Ver el contenido de un registro que ya ha añadido a un supervisor de memoria en la vista Supervisores

Procedimiento

Si es necesario, utilice las barras de desplazamiento o las teclas AvPág y RePág para desplazarse por el supervisor de memoria hasta que la dirección del registro sea visible.

Utilizar la vista Módulos

Utilice la vista de módulos para visualizar una lista de los módulos que se cargan mientras se ejecuta el programa.

Procedimiento

- Mostrar módulos con información de depuración.

  En la vista Módulos, los elementos de la lista se pueden expandir para mostrar las unidades de compilación, los archivos y las funciones. Al visualizar los módulos, pulse el botón **Mostrar módulos con información de depuración** para filtrar los módulos sin información de depuración, dejando sólo los módulos con información de depuración. De forma predeterminada, este valor está activado.

- Listar archivos de origen que están asociados con los módulos cargados.

  Al inspeccionar los módulos, al efectuar una doble pulsación en los nodos de archivos de origen se abrirán los archivos de origen en el editor. Al pulsar el botón **Mostrar diálogo de filtro de archivos** se abrirá un recuadro de diálogo con una lista de todos los archivos de código fuente asociados a los módulos cargados. Puede acotar la lista de selección escribiendo el nombre de archivo en el campo de texto. Cuando pulse **Aceptar** se abrirá el archivo de código fuente en el editor.
- Visualizar propiedades de unidades de compilación, archivos y funciones en la vista Propiedades:
  a) Para abrir la vista Propiedades, seleccione **Ventana** > **Mostrar vista** > **Propiedades**.
  b) En la vista Módulos, vaya al módulo cuyas propiedades desea visualizar. Si es necesario, expanda los nodos de módulo y utilice las barras de desplazamiento, las teclas de Arriba y Abajo o AvPág y RePág para desplazarse por la vista Módulos hasta que el módulo sea visible.
  c) Seleccione el módulo para visualizar sus propiedades en la vista Propiedades.
- Establecer puntos de interrupción de entrada desde la vista Módulos

  Pulse con el botón derecho del ratón en un punto de entrada y seleccione **Establecer punto de interrupción de entrada** en el menú.

Supervisión de memoria

Utilice la vista de memoria para cambiar el contenido de la memoria o las áreas de memoria utilizadas por el programa.

Añadir un supervisor de memoria nuevo desde la vista Variables, la vista Supervisores, la vista Registros o el editor

Procedimiento

1. En la vista Variables, la vista Supervisores o la vista Registros, pulse con el botón derecho del ratón en la variable, expresión o registro para los que desea supervisar la memoria. O bien, en el editor, resalte y pulse con el botón derecho del ratón la expresión para la que desea supervisar la memoria.

   Nota: Si la expresión es un puntero, el valor de la expresión se utilizará para direccionar la memoria. Si la expresión es un `lvalue` (con una dirección en memoria), su dirección se utilizará para direccionar la memoria. De lo contrario, el valor de la expresión se utilizará como dirección. Por ejemplo, dada la declaración `int i = 0x44;`, si la expresión es `i`, el supervisor de memoria estará en la dirección de `i`. Si la expresión es `i+1`, el supervisor de memoria estará en la ubicación proporcionada por el valor de la expresión `i+1`, que es `0x45`.
2. Seleccione **Supervisar memoria > *< rendering>*** en el menú emergente, donde *< rendering>* es la representación que desea visualizar en la parte **Representaciones** de la vista Memoria.

Añadir un nuevo supervisor de memoria para una expresión desde la vista Memoria

Procedimiento

1. Pulse **Añadir supervisor de memoria**  .
2. En el cuadro de diálogo Supervisar memoria, especifique la expresión en el campo (la expresión debe evaluarse en una dirección).
3. Pulse **Aceptar**.

   La parte **Supervisores** (izquierda) de la vista Memoria muestra la expresión que ha especificado para la supervisión. Si tiene varios supervisores de memoria, esta sección muestra una lista de expresiones que está supervisando.

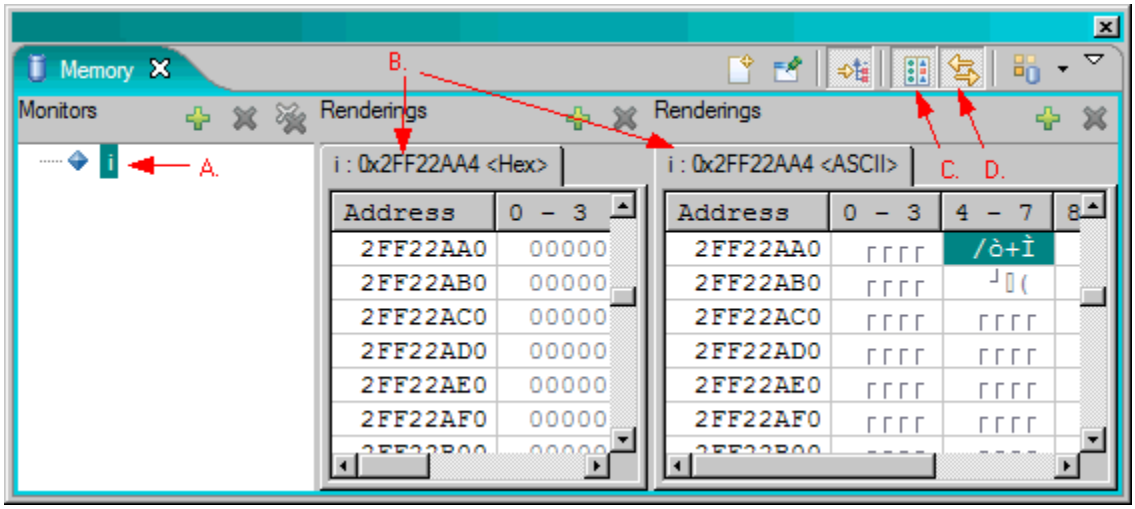
   La parte **Representaciones** (derecha) de la vista Memoria se llena con representaciones HEX y ASCII.

Inspeccionar memoria en la vista Memoria

Con la vista Memoria puede mirar el contenido de la memoria en una dirección específica. La dirección puede obtenerse de una expresión que haga referencia a una variable o a un registro. Al supervisar memoria, puede establecer que el supervisor se visualice en un formato de datos como, por ejemplo, Hex, ASCII, EBCDIC, UTF-8, entero con signo y entero sin signo.

Acerca de esta tarea

La vista Memoria se describe según este diagrama:



- El panel **Supervisores** está ubicado en el lado izquierdo de la vista (puntero A. en el diagrama). Esta parte de la vista contiene una lista de expresiones, variables y registros que ha añadido para supervisar. En esta documentación, un *supervisor* es un supervisor de memoria que se encuentra en la lista del panel **Supervisores**.
- El panel **Representaciones** está ubicado en el lado derecho de la vista (si la vista se establece en la orientación horizontal) y contiene representaciones para el supervisor seleccionado (puntero B. en el diagrama). Se utiliza para establecer el formato o formatos de datos que desea visualizar para la memoria supervisada.
- El control **Conmutar panel dividido** (puntero C. en el diagrama) permite dividir el panel **Representaciones**. De forma predeterminada, la vista Memoria sólo visualiza un panel de representación. Si pulsa **Conmutar a panel dividido**, se abrirá una segunda representación que se visualizará en forma de panel dividido.
- El control **Enlazar paneles de representación de memoria** (puntero D. en el diagrama) le permite mantener sincronizados los paneles de **Representaciones** divididos al navegar por una representación o cambiar el formato de la misma.

Al supervisar una dirección o expresión desde el panel **Supervisores**, el panel **Representaciones** se llena con la representación basada en texto predeterminada del sistema operativo. Al supervisar una dirección o expresión desde el panel **representaciones**, el panel se llena con una lista de representaciones entre las que puede elegir una para visualizarla.

Nota: Si desea visualizar memoria en formato UTF-8, utilice las representaciones **Tradicional** o **Hexadecimal y caracteres**. Para visualizar caracteres en la codificación UTF-8, pulse con el botón derecho del ratón sobe la representación **Texto** > **UTF-8** en el menú.

Puede supervisar varias variables, expresiones y registros en la vista Memoria o puede añadir varias representaciones al panel **Representaciones**. En el panel **Supervisores** aparece cada variable, expresión o registro que ha añadido. En el panel **Representaciones**, solamente se visualizan una o varias representaciones de memoria para el supervisor seleccionado actualmente en la vista **Memoria** (si hay varias representaciones se separan con pestañas o aparecen en un panel dividido).

Puede establecer los dos paneles de la vista Memoria para que se visualicen en un sentido horizontal (lado a lado) o en una orientación vertical (de arriba abajo). Para establecer el diseño de la vista, pulse el icono de flecha abajo de la vista Memoria y seleccione **Diseño** en el menú. Esto abrirá un submenú en el que puede elegir la orientación que desea visualizar.

Ver el contenido de la memoria utilizando supervisores de memoria

Acerca de esta tarea

para ver el contenido de la memoria desde la vista **Memoria**:

Procedimiento

1. En el panel **Supervisores**, seleccione el supervisor de memoria que contiene la ubicación de memoria que desea visualizar. La memoria aparecerá en el panel **Representaciones**, en el que realizará todas las demás tareas. Si ha añadido varias representaciones, seleccione la pestaña que contiene la representación que desea ver.
2. Si lo desea, divida el panel **Renderings** seleccionando el pulsador **Toggle Split Pane** (). De forma predeterminada, la vista Memoria solo muestra un panel de representación. When you click **Toggle Split Pane**, a second rendering opens and displays as a split pane. If you have chosen to render **Hex and Character**, you may need to choose this push button to see both renderings.
3. Si es necesario, utilice la barra de desplazamiento en la representación para visualizar las ubicaciones de memoria situadas por encima o por debajo de la dirección base del supervisor de memoria mostrado por la representación actual. O bien, puede pulsar el botón derecho del ratón en la representación y elegir la opción **Ir a dirección** del menú desplegable o pulsar Ctrl+G. Con ello se abrirá una sección en la parte inferior de la representación, en la que puede realizar las acciones siguientes:
   a) Seleccionar la opción **Ir a dirección** del menú desplegable y especificar una dirección a la que desea desplazarse. La representación se colocará de modo que la dirección especificada sea visible y esté seleccionada.
   b) Seleccionar la opción **Ir a desplazamiento** del menú desplegable y especificar el desplazamiento. La representación se colocará de modo que la dirección de la expresión (dirección base), más el desplazamiento especificado sea visible y esté seleccionado. Un valor negativo colocará la representación detrás de la dirección base.
   c) Seleccionar la opción **Saltar unidades de memoria** del menú desplegable. Esta función toma la dirección seleccionada actualmente y añade el número de unidades de memoria especificado. La dirección resultante se selecciona. Un valor negativo colocará la representación detrás de la dirección actual.

   Puede entrar todas estas entradas como HEX marcando el recuadro de selección **Entrar como hexadecimal** (si el recuadro no está seleccionado, la entrada será decimal). Una vez realizada la entrada en el campo, pulse Intro o pulse **Aceptar** para ir a la ubicación de la representación. Para cerrar esta sección, pulse **Cancelar** o Control+G.

   Nota: La entrada también se trata como HEX si el prefijo es `0x`.
4. Para ir a la dirección de una celda determinada, pulse dentro de la celda con el botón derecho del ratón y seleccione **Desreferenciar puntero** en el menú emergente.
5. Si lo desea, puede cambiar la anchura de cualquier columna pulsando el lado izquierdo o derecho de la celda de cabecera correspondiente y arrastrándolo para alterar la anchura de la columna o puede pulsar con el botón derecho dentro de la representación y seleccionar **Redimensionar para ajustar** en el menú emergente de modo que todas las columnas se redimensionan para que se pueda ver el texto que hay dentro de ellas. También puede pulsar con el botón derecho dentro de la representación y seleccionar **Formatear** en el menú emergente. Esto abrirá el recuadro de diálogo Formato. En este recuadro de diálogo, puede establecer el número de unidades por fila y el número de unidades por columna. A medida que establezca estos valores, una ventana **Vista previa** del recuadro de diálogo

visualizará el diseño de representación que está estableciendo. Para guardar estos valores como diseño predeterminado, pulse **Guardar como valores predeterminados**.

6. Para conmutar la representación de memoria a *Modalidad de desplazamiento*, pulse con el botón derecho dentro de la representación y seleccione **Cambiar modalidad de visualización > Modalidad de desplazamiento** en el menú emergente. Para conmutar la representación de memoria a *Modalidad de dirección*, pulse con el botón derecho dentro de la representación y seleccione **Cambiar modalidad de visualización > Modalidad de dirección** en el menú emergente. Cuando cambia a Modalidad de desplazamiento, la dirección de la expresión que se está supervisando se vuelve la primera celda de la representación y la columna **Dirección** visualiza desplazamientos.
7. También puede ocultar elementos de la vista memoria para facilitar la visualización:
   - Puede ocultar el panel **Supervisores** deseleccionando el conmutador **Conmutar panel de supervisores de memoria**.
   - Puede ocultar la columna **Dirección** pulsando con el botón derecho del ratón dentro de la representación y seleccionando **Ocultar columna Dirección**. Para restaurar la columna de dirección donde está oculta, pulse con el botón derecho sobre la representación y seleccione **Mostrar columna de dirección** en el menú emergente.

Resultados

Si está en una representación de memoria y sale de la dirección establecida originalmente para supervisar, al elegir el elemento de menú emergente **Restablecer en dirección base** el cursor se volverá a colocar en la dirección base del supervisor de memoria. Como alternativa, puede restablecer todas las representaciones de un supervisor de memoria pulsando el supervisor con el botón derecho del ratón y seleccionando **Restablecer** (o bien puede seleccionar varios supervisores y elegir esta acción). Al restablecer un supervisor, de forma predeterminada, las representaciones visibles se restablecerán en la dirección base. Para restablecer todas las representaciones de la vista Memoria actual en la dirección base, modifique las preferencias de la vista Memoria.

Cambiar el contenido de una ubicación de memoria

Al depurar, puede cambiar el contenido de una ubicación de memoria.

Acerca de esta tarea

Para cambiar el contenido de una ubicación de memoria en un supervisor de memoria de la vista Memoria:

Procedimiento

1. En el panel **Supervisores**, seleccione el supervisor de memoria que contiene la ubicación de memoria que desea editar. La memoria aparecerá en el panel **Representaciones**, en el que realizará todas las demás tareas. Si ha añadido varias representaciones, seleccione la pestaña que contiene la representación que desea editar.
2. Desplácese hasta la ubicación de memoria que desea cambiar. También puede pulsar el supervisor con el botón derecho del ratón y elegir la opción **Ir a dirección** del menú emergente. Con ello abrirá una sección **Ir a dirección** en la parte inferior de la representación, en la que puede especificar una dirección a la que desea desplazarse.
3. Seleccione la fila que contiene el valor que desea cambiar y efectúe una doble pulsación sobre el valor que desea cambiar.

   Consejo: Si la representación está actualmente en primer plano, no es necesario efectuar una doble pulsación sobre el valor que desea cambiar para poder editarlo. En lugar de ello, puede simplemente empezar a escribir el cambio y el editor se activará.
4. Especifique un valor válido para esa ubicación de memoria.
5. Pulse **Intro** para someter el cambio. El depurador comprobará la validez del valor.

Resultados

Preferencias de la vista Memoria

Puede establecer preferencias de representación de tabla, página de códigos y serie de relleno para las representaciones de memoria. Además, puede modificar el comportamiento preferido para restablecer las representaciones de memoria.

Acerca de esta tarea

Los recuadros de diálogo de preferencias de la vista Memoria se abren desde el menú del icono de flecha abajo de la vista Memoria. Para abrir el recuadro de diálogo Preferencias de la vista Memoria, pulse el icono de flecha abajo de la vista Memoria y seleccione **Preferencias** en el menú. Para abrir el recuadro de diálogo Preferencias de las representaciones de tabla de la vista Memoria, pulse el icono de flecha abajo de la vista Memoria y seleccione **Preferencias de representaciones de tabla** en el menú.

Para restaurar los cambios efectuados en las preferencias en sus valores predeterminados, pulse **Restaurar valores predeterminados**.

Preferencias: Restablecer supervisor de memoria

Acerca de esta tarea

Puede restablecer una representación en la dirección base si la ha movido de ella. Al restablecer una representación en la dirección base, puede establecerla para restablecer sólo la representación visible o para restablecer todas las representaciones. Si elige restablecer todas las representaciones, el rendimiento de la operación de restablecimiento puede resultar afectado negativamente. Para establecer esta preferencia, abra el recuadro de diálogo Preferencias y seleccione el nodo **Restablecer supervisor de memoria**. En la página Restablecer supervisor de memoria, elija el pulsador adecuado.

Preferencias: Serie de relleno

Acerca de esta tarea

La serie de relleno es la serie que aparecerá en el contenido de la memoria cuando ésta no pueda recuperarse. Para establecer la serie de relleno, abra el recuadro de diálogo Preferencias y seleccione el nodo **Serie de relleno**. En la página Serie de relleno, especifique la serie que desea visualizar cuando no pueda determinarse el contenido de la memoria.

Preferencias: Seleccionar páginas de códigos

Acerca de esta tarea

Al supervisar representaciones basadas en texto ASCII y EBCDIC (y memoria correlacionada, si no está disponible en el producto con el que instaló este depurador) en el panel **Representaciones**, puede establecer la página de códigos en el que desea visualizar la representación.

Para establecer la página de códigos que representará la memoria en ASCII/EBCDIC, abra el recuadro de diálogo Preferencias y seleccione el nodo **Seleccionar páginas de códigos**. En la página Seleccionar páginas de códigos, especifique la página de códigos del juego de caracteres que desea cambiar (para representaciones ASCII, EBCDIC o ambas).

Preferencias de representaciones de tabla

Acerca de esta tarea

Para establecer las preferencias de las representaciones de tabla que se visualizan en una tabla, pulse el icono de flecha abajo de la vista Memoria y seleccione **Preferencias de representaciones de tabla**. En el recuadro de diálogo de preferencias que se abre, puede indicar si desea que el depurador cargue automáticamente la próxima página de memoria cuando el usuario se desplaza hacia el final del

almacenamiento intermedio. Si deselecciona este valor, el número de líneas por página que especifique se cargará en el panel **Representaciones**. No podrá desplazarse fuera del almacenamiento intermedio definido por este valor de tamaño de página. En lugar de ello, para visualizar la memoria de la página siguiente o anterior, debe pulsar el botón derecho del ratón para utilizar las acciones **Página anterior** y **Página siguiente** del menú emergente.

Resultados

Trabajar con varias vistas Memoria

Acerca de esta tarea

You can add additional Memory views to the workbench. Para ello, pulse **Nueva vista de memoria** (). Cuando tiene varias vistas de memoria abiertas, no puede enlazar sus representaciones entre sí. However, you can pin the contents of a Memory view so that memory renderings that are added to one view do not affect the other view. Para fijar un supervisor de memoria, asegúrese de que el botón **Fijar supervisor de memoria** () en la vista Memoria esté activado. If you then go to another Memory view and add a memory monitor, it will show up in both Memory views, however, the memory rendering that is currently displayed in the pinned monitor will not change.

Cuando añada una vista Memoria nueva, su panel **Representaciones** se llenará con la lista de selección de representaciones de memoria. En esta lista, puede seleccionar el formato de datos que desea utilizar para la representación de memoria y pulsar **Añadir representaciones**.

Eliminar supervisores de memoria de la vista Memoria

Acerca de esta tarea

Para eliminar un supervisor de memoria de la vista **Memoria**:

Procedimiento

1. Seleccione el supervisor de memoria que desea eliminar (seleccionándolo en la lista en el panel **Supervisores**).
2. Pulse el pulsador **Eliminar supervisor de memoria** ().

Resultados

Para eliminar varios supervisores de memoria, selecciónelos mediante las teclas Control o Mayús y pulse **Eliminar supervisor de memoria**. Para eliminar todos los supervisores de memoria, pulse **Eliminar todo**.

Nota: Si ha añadido varias representaciones para un supervisor de memoria, todas ellas se eliminarán cuando elija eliminar el supervisor.

Correlacionar memoria

En la vista Memoria, puede visualizar el contenido de la memoria correlacionada según un diseño definido por el usuario o según un diseño de ejemplo.

Acerca de esta tarea

Dependiendo del producto que esté ejecutando, puede haber diseños de ejemplo y/o archivos DTD (definición de tipo de documento) disponibles en estas ubicaciones:

- En `<directorio de instalación del producto> \plugins\com.ibm.debug.memorymap.<plataforma>.samples\samples`, donde `<directorio de instalación del producto>` es el directorio donde ha instalado este producto.
- En `&lt;directorio de instalación del producto>\maps`.

Los diseños de memoria predefinidos se almacenan en archivos XML (un archivo XML para cada diseño, creado mediante un editor de texto o XML). El formato de archivo XML proporciona estructuras descriptivas de elementos de tipo primitivo predefinidos o diseños anidados, donde un elemento de diseño puede señalar a otro archivo de diseño de memoria. El archivo de diseño también especifica la longitud del bloque de memoria que debe diseñarse.

El tamaño de un bloque de memoria que se supervisa está determinado por el tamaño del diseño seleccionado. Si el bloque de memoria especificado está protegido o no se puede acceder a él, los valores de visualización se mostrarán como la serie que se establece como la serie rellenada en las preferencias de usuario (de forma predeterminada, varios "?").

Inicialmente, el elemento de diseño y los subelementos que representan diseños anidados no se llenan (aún no se generan subelementos). La primera vez que expanda un elemento de diseño, éste se llenará de acuerdo con el archivo de diseño XML. Llenar el elemento de diseño significa romper el bloque de memoria en fragmentos correspondientes a los elementos de diseño especificados en el archivo XML. Los valores visualizados para los subelementos de diseño se formatean de acuerdo con un tipo primitivo predeterminado especificado en el archivo XML.

Utilización con memoria correlacionada
Puede utilizar la vista Memoria para supervisar expresiones, variables y registros de la memoria por correlación de memoria.

Acerca de esta tarea

Para ver memoria correlacionada en un supervisor que ha añadido a la vista Memoria:

Procedimiento

1. Defina columnas en el panel **Representaciones** de la vista Memoria. Puede mostrar u ocultar columnas pulsando con el botón derecho del ratón en el interior del panel y seleccionando **Elegir columnas** en el menú emergente. En el recuadro de diálogo resultante, seleccione las columnas que desea mostrar y pulse **Aceptar**. También puede mover columnas en la vista, arrastrándolas y soltándolas.
2. Si es necesario, utilice la barra de desplazamiento de la representación para ver los campos. También puede pulsar el supervisor con el botón derecho del ratón y elegir la opción **Buscar campo** del menú emergente. Para obtener más información acerca de cómo buscar campos, consulte el tema relacionado.
3. Si lo desea, establezca la representación para que muestre u oculte los tipos. Para ello, pulse con el botón derecho del ratón el supervisor de correlación de memoria y seleccione **Mostrar tipos** u **Ocultar tipos** en el menú emergente.
4. Si lo desea, cambie la representación del contenido de la memoria para el campo que está visualizando. Para ello, pulse con el botón derecho del ratón sobre el campo o su valor y seleccione **Representación > <formato de representación>** en el menú emergente.
5. Elija el tipo de visualización deseado para los valores de la columna **Desplazamiento** pulsando con el botón derecho del ratón en el panel **Representaciones** y seleccionando **Elegir visualización de desplazamiento** > **tipo de visualización de desplazamiento**.
6. Para facilitar la visualización, puede agrupar los campos de memoria y establecer filtros para estos grupos. Para obtener información acerca de la agrupación de campos de diseño de correlación, consulte el tema relacionado.

Resultados

Puede supervisar varias variables, expresiones y registros en la vista Memoria o añadir varias representaciones de correlación de un único supervisor de memoria. También puede añadir varias vistas Memoria adicionales al entorno de trabajo. En el panel **Supervisores** de la vista Memoria se lista cada una de las variables, expresiones o registros que ha añadido. En el panel **Representaciones**, sólo se visualizan las representaciones de memoria correspondientes al supervisor seleccionado actualmente en la vista **Memoria** (las diversas representaciones se separan mediante pestañas o un panel dividido).

Establecer preferencias de correlación de memoria
En las preferencias de correlación de memoria, puede establecer la ubicación de la correlación de memoria. Además, puede indicar si desea que el depurador le solicite confirmación cuando elige eliminar todos los grupos al trabajar en el diálogo Gestionar grupos. También puede establecer la correlación para que se construya antes de buscar campos.

Acerca de esta tarea

El producto con el que ha instalado el depurador puede incluir un directorio de correlación de memoria de ejemplo de `<product installation directory>\plugins\com.ibm.debug.memorymap.<platform>.samples\samples`, donde `<product installation directory>` es el directorio en el que ha instalado este producto. Si el producto incluye este directorio, el depurador buscará en él las correlaciones de memoria de forma predeterminada. De lo contrario, el directorio de correlación de memoria predeterminado puede encontrarse en las preferencias de correlación de memoria. El directorio de correlación de memoria debe contener un archivo `layout.dtd`, que es necesario para la vista Memoria. Puede cambiar la ubicación de correlación de memoria, pero si lo hace debe copiar un archivo `layout.dtd` en la nueva ubicación de correlación de memoria (si exporta una correlación a esta ubicación, el procedimiento de exportación generará automáticamente un archivo `layout.dtd`). Este archivo debe residir siempre en la ubicación de correlación de memoria.

Nota: Puede que también haya un archivo `layout.dtd` disponible en el sitio de descarga del producto con el que ha instalado este depurador. Si no hay un archivo `layout.dtd` disponible en el producto con el que ha instalado este depurador, puede crear un archivo `layout.dtd` según se describe en el tema "Definir un diseño de correlación" en la página 367.

Para que el depurador busque correlaciones de memoria creadas por el usuario, puede añadir las correlaciones de memoria al directorio predeterminado o cambiar la ubicación de las correlaciones de memoria de forma que señale a otro directorio, del siguiente modo (asegúrese de que este otro directorio contiene una copia del archivo `layout.dtd`):

Procedimiento

1. En la vista Memoria, pulse el icono de flecha abajo y seleccione **Preferencias de correlación de memoria** en el menú.
2. En el recuadro de diálogo **Preferencias de correlación de memoria**, especifique o busque la ubicación de correlación de memoria que desea establecer en el campo **Ubicación de correlaciones de memoria**.

   Nota:

   - Si el producto con el que ejecuta este depurador se suministra con el Explorador de sistemas remotos, los valores de ubicación de los mapas de memoria se realizan en este recuadro de diálogo en la sección **Ubicación de correlaciones de memoria**. En esta sección, puede especificar o buscar una ubicación en un servidor remoto. Para ello, elija el **Perfil** y la **Conexión** asociados con la ubicación de correlación de memoria (si no especifica un perfil o no existe ninguno en el espacio de trabajo, el nombre de archivo especificado en el campo **Directorio** se tratará como un archivo local y no se asociará con ningún perfil). A continuación, especifique la carpeta de ubicación de correlación de memoria en el campo **Directorio**. Al correlacionar memoria, visualizará una lista de las correlaciones que residen en la ubicación especificada. Si esta ubicación es remota, se intentará una conexión con el servidor remoto para recuperar la lista de correlaciones disponibles. Si se selecciona la opción **Correlacionar**, ésta le permitirá buscar una correlación tanto en el sistema local como en los remotos. Si el archivo de correlación seleccionado se encuentra en un sistema remoto, los archivos remotos necesarios se almacenarán en la memoria caché del sistema local.
   - Si ha modificado la ubicación de la correlación de memoria, puede recuperar fácilmente el valor predeterminado del producto pulsando en el recuadro de diálogo **Preferencias de correlación de memoria** el pulsador **Restaurar valores predeterminados**.
3. Si desea controlar el tamaño del bloque de memoria que se recupera, complete los campos **Tamaño mínimo de recuperación de bloque de memoria en bytes** y **Tamaño máximo de recuperación de**

bloque de memoria en bytes. Cuando se recupera un bloque de memoria, se divide en segmentos con un tamaño igual al tamaño mínimo de recuperación de bloque de memoria. Las solicitudes de recuperación se consolidan hasta el tamaño máximo de recuperación de bloque de memoria.

Nota:

- Si el tamaño máximo de recuperación de bloque de memoria especificado supera el tamaño máximo soportado por el motor del depurador, se utilizará el tamaño máximo soportado por el motor del depurador.
- Si observa problemas de rendimiento en la correlación de memoria, el aumento del tamaño mínimo de bloque puede ser de ayuda. En correlaciones grandes y contiguas, un valor más grande del tamaño mínimo de bloque aumentará el rendimiento.

4. Seleccione el recuadro de selección **Solicitar al eliminar todos los grupos** si desea recibir una solicitud cuando elimine todos los grupos.
5. Elija si desea o no recibir una solicitud para conservar o descartar información de agrupación y descripción antes de reconstruir una correlación. Si no se selecciona este recuadro de selección, se recordará la última acción de guardado o descarte (por ejemplo, se guardará la información si era para la última reconstrucción de correlación).
6. Indique si desea que el archivo de correlación XML se guarde cuando se modifiquen los grupos y descripciones en la representación. Si se selecciona este recuadro de selección, la representación se reconstruirá cuando efectúe cambios, y se reconstruirán las representaciones de la vista Memoria que utilicen el archivo XML relacionado.
7. Para construir la correlación antes de abrir el recuadro de diálogo Buscar campo, marque el recuadro de selección **Construir automáticamente la correlación antes de abrir el diálogo Buscar campo**. Si no se marca este recuadro de selección, sólo aquellos elementos que ya se han construido (o expandido) en la correlación se visualizarán en el recuadro de diálogo Buscar campo. De forma predeterminada, este recuadro de selección está seleccionado.
8. Especifique el valor elegido para recibir un mensaje de aviso cuando la exportación de una correlación afecte a otras representaciones de memoria.

Resultados

Al correlacionar memoria, la lista de correlaciones disponibles que se visualiza corresponde a correlaciones residentes en la ubicación de correlación de memoria. De forma similar, al correlacionar memoria mediante la acción **Correlacionar**, se le solicitará que localice la correlación en esta ubicación; sin embargo, con esta acción también puede buscar las correlaciones de memoria en otras ubicaciones del sistema local. Si efectivamente busca en otras ubicaciones del sistema local y elige una correlación de esta ubicación, la ubicación se convertirá en la ubicación de correlación de memoria predeterminada.

Nota: Si el producto con el que ejecuta este depurador se suministra con el Explorador de sistemas remotos, puede buscar un mapa en un sistema remoto o local. Si elige una correlación de una ubicación diferente de un sistema local o remoto, la ubicación se convertirá en la ubicación de correlación de memoria predeterminada.

Correlacionar memoria para una expresión, variable o registro

Para correlacionar memoria para una expresión o variable, siga las instrucciones destinadas a añadir una expresión o variable a la vista Memoria y, a continuación, elija la opción **Correlacionar** al seleccionar la representación de memoria. De forma similar, para correlacionar memoria para un registro, siga las instrucciones destinadas a añadir un registro a la vista Memoria y elija la opción **Correlacionar** al seleccionar la representación de memoria.

Acerca de esta tarea

Para obtener información acerca de cómo añadir expresiones, variables y registros a la vista Memoria, consulte los temas relacionados.

Si elige visualizar memoria con una correlación que contiene errores, el panel **Representaciones** de la vista Memoria visualizará un mensaje de error que contiene opciones para resolverlo. Por ejemplo, la página de error puede incluir opciones para abrir el archivo (lo cual le permitirá editarlo y guardarlo)

y para reconstruir el archivo. Al correlacionar memoria, la correlación sólo construye elementos para nodos expandidos. Puede que no encuentre errores hasta que se expanda el nodo que contiene un error. Para solucionar estos errores, abra la correlación, solucione los errores y, a continuación, reconstruya la correlación. Para obtener información acerca de cómo editar diseños de memoria, consulte el tema relacionado.

Definir un diseño de correlación
La siguiente información describe la definición de diseño de una correlación con un ejemplo.

Acerca de esta tarea

Crear el archivo XML de diseño

El formato del archivo XML se define en el archivo DTD (definición de tipo de documento) layout.dtd del siguiente modo:

```
<?xml version="1.0"?>
  <!ELEMENT LAYOUT (FIELD)+>
  <!ATTLIST LAYOUT Header CDATA #REQUIRED length CDATA #REQUIRED>
  <!ELEMENT GROUP EMPTY>
  <!ATTLIST GROUP Name CDATA #REQUIRED>
  <!ELEMENT FIELD (FIELD)*>
  <!ATTLIST FIELD
    Header CDATA #REQUIRED
    Type (16_BIT_INT|16_BIT_UINT|16_BIT_HINT|32_BIT_INT|32_BIT_UINT|32_BIT_HINT|32_BIT_FLOAT|
64_BIT_INT|64_BIT_FLOAT|CHARACTER|HEX|ASCII|EBCDIC|STRUCTURE|PADDING|BIT|BITMASK|MAP) #REQUIRED
    length CDATA #REQUIRED
    layout CDATA #IMPLIED
    filename CDATA #IMPLIED
      Groups CDATA #IMPLIED>
```

Esto significa que el archivo de diseño XML especifica primero una cabecera (title) y la longitud total del diseño, seguido de una lista de subelementos (FIELD) descritos por una cabecera (name), la longitud y el tipo primitivo que se utilizan para determinar la representación predeterminada de ese subelemento.

También hay tipos de subelementos especiales:

- PADDING, se utiliza para definir un bloque de bytes que no necesita diseñarse específicamente
- STRUCTURE introduce una estructura anidada; el subelemento no tiene valor
- BITMASK, se utiliza para definir un subelemento con máscara de bits. Sus subelementos representan bits, grupos o bits definidos por el tipo BIT.
- UNION define la misma parte de memoria de más de una forma.

El ejemplo siguiente define el diseño de la siguiente estructura de lenguaje C:

```
typedef struct {
  unsigned short ushort_val;
  short short_val;
  unsigned long ulong_val;
  long long_val;
  char string_val[12];
  char char_val;
} _test;
```

El archivo XML que describe una vista de árbol de la estructura `_test` conforme a este formato es:

```
<?xml version="1.0"?>
  <LAYOUT Header="A Layout" description="Tree view" length="25">
  <FIELD Header="ushort_val" Type="16_BIT_UINT" length="2"></FIELD>
  <FIELD Header="short_val" Type="16_BIT_INT" length="2"></FIELD>
  <FIELD Header="ulong_val" Type="32_BIT_UINT" length="4"></FIELD>
  <FIELD Header="long_val" Type="32_BIT_INT" length="4"></FIELD>
  <FIELD Header="string_val" Type="ASCII" length="12"></FIELD>
  <FIELD Header="char_val" Type="ASCII" length="1"></FIELD>
</LAYOUT>
```

Los atributos `offset` y `offset_mode` le permiten especificar la ubicación exacta de un campo en relación con el inicio de la correlación (`offset_mode=absolute`) o en relación con la dirección actual

(offset_mode=relative). En el ejemplo siguiente, el elemento llamado b tiene un offset de 10 y offset_mode se define como relative. Sin estos atributos, este elemento tendría un desplazamiento de 80, pero debido a que el desplazamiento se define como 10, en relación con la posición actual, el desplazamiento es 90. Observe que la longitud del elemento a se define en hexadecimal porque la longitud tiene el prefijo 0x. Generalmente, los atributos length y offset se pueden especificar en HEX con el prefijo 0x. El elemento c tiene un desplazamiento de 4 y la modalidad es absolute. Esto significa que el desplazamiento de este elemento es 4 bytes contados a partir del inicio del diseño. Los 10 bytes correlacionados por el campo c también los cubre el campo a:

```
<Header="offset_Test" length="190">
  <FIELD Header="a" Type="HEX" length="0x64"></FIELD>
  <FIELD Header="b" description="offset = 90" Type="HEX" length="80" offset="10"
offset_mode="relative"></FIELD>
  <FIELD Header="c" Type="HEX" length="10" offset="4" offset_mode="absolute"></FIELD>
</LAYOUT>
```

Definir campos de relleno

El relleno de los campos se puede utilizar para manejar estructuras alineadas de byte o para saltar áreas de datos en la correlación que no es necesario definir en la correlación de memoria. Por ejemplo, para la estructura _test definida más arriba, puede crear una correlación que ignora el campo long_val pero que muestra el tipo string_val en el diseño. El archivo XML tendría el aspecto siguiente:

```
<?xml version="1.0"?>
  <LAYOUT Header="A Layout" description="Tree view" length="0x19">
  <FIELD Header="ushort_val" Type="16_BIT_UINT" length="2"></FIELD>
  <FIELD Header="short_val" Type="16_BIT_INT" length="2"></FIELD>
  <FIELD Header="ulong_val" Type="32_BIT_UINT" length="4"></FIELD>
  <FIELD Header="" Type="PADDING" length="4"></FIELD>
  <FIELD Header="string_val" Type="ASCII" length="12"></FIELD>
  <FIELD Header="char_val" Type="ASCII" length="1"></FIELD>
</LAYOUT>
```

También podría utilizar aquí el atributo offset para omitir el campo long_val, especificando el offset de string_val como 4, y offset_mode como relative:

```
  <FIELD Header="string_val" Type="ASCII" length="12" offset="4" offset_mode="relative"></FIELD>
```

Esto significa que la dirección del campo string_val es realmente 4 bytes relativos último byte del campo ulong_val, saltándose así los bytes utilizados por el campo long_val.

Definir estructuras

El siguiente ejemplo XML muestra la utilización de campos STRUCTURE para correlacionar estructuras anidadas. Un elemento superior de estructura no tiene un valor asociado y puede expandirse para mostrar sus subelementos. Aunque la longitud del campo STRUCTURE se añade al tamaño total del diseño XML, los tamaños de los campos incluidos sólo están destinados a visualización. Por ejemplo, la estructura siguiente indica sólo 344 bytes del tamaño total del diseño.

```
<FIELD Header="MACHINE CHECK LOG OUT AREA" Type="STRUCTURE" length="344">
  <FIELD Header="reserved" Type="HEX" length="16"></FIELD>
  <FIELD Header="FLCSID" Type="HEX" length="4"></FIELD>
  <FIELD Header="FLCIOFP" Type="HEX" length="4"></FIELD>
  <FIELD Header="reserved" Type="HEX" length="20"></FIELD>
  <FIELD Header="FLCESAR" Type="HEX" length="4"></FIELD>
  <FIELD Header="FLCCTSA" Type="HEX" length="8"></FIELD>
  <FIELD Header="FLCCCSA" Type="HEX" length="8"></FIELD>
  <FIELD Header="FLCMCIC" Type="HEX" length="8"></FIELD>
  <FIELD Header="reserved" Type="HEX" length="8"></FIELD>
  <FIELD Header="FLCFSA" Type="HEX" length="4"></FIELD>
  <FIELD Header="reserved" Type="HEX" length="4"></FIELD>
  <FIELD Header="FLCFLA" Type="HEX" length="16"></FIELD>
  <FIELD Header="FLCRV110" Type="HEX" length="16"></FIELD>
  <FIELD Header="FLCARSAV" Type="STRUCTURE" length="64">
    <FIELD Header="AR0" Type="HEX" length="4"></FIELD>
    <FIELD Header="AR1" Type="HEX" length="4"></FIELD>
    <FIELD Header="AR2" Type="HEX" length="4"></FIELD>
    <FIELD Header="AR3" Type="HEX" length="4"></FIELD>
```

```
    <FIELD Header="AR4" Type="HEX" length="4"></FIELD>
    <FIELD Header="AR5" Type="HEX" length="4"></FIELD>
    <FIELD Header="AR6" Type="HEX" length="4"></FIELD>
    <FIELD Header="AR7" Type="HEX" length="4"></FIELD>
    <FIELD Header="AR8" Type="HEX" length="4"></FIELD>
    <FIELD Header="AR9" Type="HEX" length="4"></FIELD>
    <FIELD Header="AR10" Type="HEX" length="4"></FIELD>
    <FIELD Header="AR11" Type="HEX" length="4"></FIELD>
    <FIELD Header="AR12" Type="HEX" length="4"></FIELD>
    <FIELD Header="AR13" Type="HEX" length="4"></FIELD>
    <FIELD Header="AR14" Type="HEX" length="4"></FIELD>
    <FIELD Header="AR15" Type="HEX" length="4"></FIELD>
  </FIELD>
  <FIELD Header="FLCFPSAV" Type="HEX" length="32"></FIELD>
  <FIELD Header="" Type="PADDING" length="64"></FIELD>
  <FIELD Header="" Type="PADDING" length="64"></FIELD>
</FIELD>
```

Las estructuras pueden definirse interna o externamente para un diseño. Se puede crear una estructura externa como un diseño anidado especificando `filename="<nombre de archivo>"` en el campo de estructura, donde el archivo al que hace referencia `<nombre de archivo>` contiene la definición real de la estructura.

Por ejemplo, la estructura `MACHINE CHECK LOG OUT AREA` se puede especificar en un diseño de correlación externamente de la siguiente manera: `&lt;FIELD Header="MACHINE CHECK LOG OUT AREA" Type="STRUCTURE" length="344" filename="machine.xml">&lt;/FIELD>`.

Definir campos de máscaras de bits

El fragmento de XML que sigue es un ejemplo para describir campos BITMASK. La longitud de BITMASK se especifica en bytes y contiene un conjunto de campos BIT para los que la longitud se especifica en bits. El desplazamiento mostrado para los campos BIT es un desplazamiento de bits dentro del campo BITMASK. Aunque la longitud del campo de máscara de bits se añade al tamaño total del diseño XML, los tamaños de los campos BIT individuales sólo están destinados a visualización.

```
<FIELD Header="BITMASK" Type="BITMASK" length="1">
  <FIELD Header="BIT 1" Type="BIT" length="1"></FIELD>
  <FIELD Header="BIT 2" Type="BIT" length="1"></FIELD>
  <FIELD Header="BIT 3" Type="BIT" length="1"></FIELD>
  <FIELD Header="BIT 4" Type="BIT" length="1"></FIELD>
  <FIELD Header="BIT 5" Type="BIT" length="1"></FIELD>
  <FIELD Header="BIT 6" Type="BIT" length="1"></FIELD>
  <FIELD Header="BIT 7" Type="BIT" length="1"></FIELD>
  <FIELD Header="BIT 8" Type="BIT" length="1"></FIELD>
</FIELD>
```

Definir uniones

El ejemplo siguiente define el diseño de la siguiente unión de lenguaje C:

```
union my_union {
  int my_intVal;
  double my_doubleVal;
};
```

El siguiente ejemplo de XML muestra la forma en que se describe la unión. Observe que en el XML, la longitud de la unión es el tamaño de su campo más grande:

```
<LAYOUT Header="UNIONS" length="8">
  <FIELD Header="my_union" Type="UNION" length="8">
    <FIELD Header="my_intVal" Type="HEX" length="4" description="value within the union"></
FIELD>
    <FIELD Header="my_doubleVal" Type="HEX" length="8"></FIELD>
  </FIELD>
</LAYOUT>
```

Definir diseños anidados

Utilizando conjuntamente el tipo de campo MAP y campos de diseño opcionales, puede describir diseños anidados como en el siguiente ejemplo de diseño DSA:

```
<?xml version="1.0"?>
<!DOCTYPE LAYOUT SYSTEM "Layout.dtd">
<LAYOUT Header="DSA" length="72">
  <FIELD Header="FLAGS" Type="HEX" length="2"></FIELD>
  <FIELD Header="junk" Type="HEX" length="2"></FIELD>
  <FIELD Header="Back Chain" Type="MAP" length="4" layout="dsa.xml"></FIELD>
  <FIELD Header="Forward Chain" Type="MAP" length="4" layout="dsa.xml"></FIELD>
  <FIELD Header="R14" Type="HEX" length="4"></FIELD>
  <FIELD Header="R15" Type="HEX" length="4"></FIELD>
  <FIELD Header="R0" Type="HEX" length="4"></FIELD>
  <FIELD Header="R1" Type="HEX" length="4"></FIELD>
  <FIELD Header="R2" Type="HEX" length="4"></FIELD>
  <FIELD Header="R3" Type="HEX" length="4"></FIELD>
  <FIELD Header="R4" Type="HEX" length="4"></FIELD>
  <FIELD Header="R5" Type="HEX" length="4"></FIELD>
  <FIELD Header="R6" Type="HEX" length="4"></FIELD>
  <FIELD Header="R7" Type="HEX" length="4"></FIELD>
  <FIELD Header="R8" Type="HEX" length="4"></FIELD>
  <FIELD Header="R9" Type="HEX" length="4"></FIELD>
  <FIELD Header="R10" Type="HEX" length="4"></FIELD>
  <FIELD Header="R11" Type="HEX" length="4"></FIELD>
  <FIELD Header="R12" Type="HEX" length="4"></FIELD>
</LAYOUT>
```

Este diseño XML de formato correcto se almacena en un archivo denominado DSA.XML. Dado que el usuario sabe que los campos 3 y 4 contienen punteros a estructuras DSA diferentes, añade dos definiciones de diseño anidadas.

Nota: La correlación de memoria real para ese diseño se ejecuta solamente cuando se expande el elemento de diseño por primera vez para evitar expansiones de diseño recursivas.

Definir grupos

Con la sintaxis de grupos, puede organizar los campos de los diseños de correlación en grupos para facilitar el trabajo con ellos. Para definir un grupo, debe colocar `&lt;GROUP Name="groupName">&lt;/GROUP>` en la parte superior del archivo de diseño. A continuación, indica que el campo pertenece al grupo predefinido especificando: `&lt;FIELD Header="RESERVED" Type="HEX" length="12" Groups="groupName">&lt;/FIELD>`.

Un campo puede pertenecer a varios grupos. Para definir varios grupos, especifíquelos en una lista delimitada por comas en el atributo `Groups`. Cada grupo del atributo `Groups` debe haberse definido en el diseño utilizando el código `&lt;GROUP>`.

El nombre de grupo `ALL` corresponde a un grupo especial. Si especifica este nombre en un campo, éste pertenecerá a todos los grupos y será visible en todos ellos. El ejemplo de código que sigue contiene grupos:

```
<?xml version="1.0"?>
<!DOCTYPE LAYOUT SYSTEM "Layout.dtd">
<LAYOUT Header="GROUP_EXAMPLE" length="32">
  <GROUP Name="GroupA"></GROUP>
  <GROUP Name="GroupB"></GROUP>
  <FIELD Header="FIELD_A" Type="HEX" length="8" Groups="GroupA"></FIELD>
  <FIELD Header="FIELD_B" Type="HEX" length="8" Groups="GroupB"></FIELD>
  <FIELD Header="FIELD_AB" Type="HEX" length="8" Groups="GroupA,GroupB"></FIELD>
  <FIELD Header="FIELD_ALL" Type="HEX" length="8" Groups="ALL"></FIELD>
</LAYOUT>
```

Definir grupos ORG

Puede utilizar el código `ORG_GROUP` para definir el diseño de una parte de memoria definida anteriormente. Es similar al comportamiento de la instrucción `ORG` en ensamblador. Puede especificar la ubicación inicial del nuevo diseño utilizando el atributo `FIELD`. En el caso más sencillo, el valor del atributo `FIELD` puede ser el nombre de un campo de la correlación definido anteriormente. También

puede utilizar *NONE o * como valores, lo que significa que el diseño es para la ubicación actual en memoria. El atributo Header es simplemente un nombre para el nuevo diseño.

```
<LAYOUT Header="SW00SR" length="271">
   <ORG_GROUP FIELD="*NONE" Header="ORG_GROUP1">
    <FIELD Header="A" length="4" Type="HEX"></FIELD>
    <FIELD Header="B" length="4" Type="HEX"></FIELD>
    <FIELD Header="c" length="4" Type="HEX"></FIELD>
  </ORG_GROUP>
   <ORG_GROUP FIELD="A" Header="my_custom_header">
    <FIELD Header="F" length="4" Type="HEX" description="address of F == address of A"></FIELD>
    <FIELD Header="G" length="4" Type="HEX"></FIELD>
    <FIELD Header="H" length="4" Type="HEX"></FIELD>
    <ORG_GROUP FIELD="*+4" Header="another_org">
      <FIELD Header="J" length="2" Type="HEX" description="address of J = current location +
4"></FIELD>
    </ORG_GROUP>
  </ORG_GROUP>
  <FIELD Header="R" length="4" Type="HEX"></FIELD>
  <FIELD Header="Z" length="4" Type="HEX"></FIELD>
</LAYOUT>
```

donde:

- El atributo Header es un nombre para el grupo definido y es parecido al atributo Header para los elementos de estructura o de correlación.
- El valor de FIELD se evalúa y se utiliza como dirección inicial del ORG_GROUP. Por ejemplo,
  - FIELD="*NONE" o FIELD="*" significa la ubicación actual en la correlación.
  - FIELD="* +/- a +/- b ..." también es válido, donde a y b son nombres de campos en la correlación (este elemento ya se debe haber definido) o a y b podrían ser enteros.
  - FIELD="NAME" significa la dirección del elemento de la correlación llamado NAME. Esto también puede ser una expresión, por ejemplo, FIELD="NAME" o FIELD="NAME +/- a_1 +/- a_2 .... +/- a_n", donde cada a_i es el nombre de un campo en la correlación (este elemento ya se debe haber definido) o un entero.

Editar diseños de memoria

Acerca de esta tarea

Puede editar diseños de memoria de dos maneras diferentes desde la vista Memoria. Puede establecer grupos para la correlación actual que está utilizando para la representación y, a continuación, exportar la correlación (esta acción sobrescribirá la correlación existente si exporta la correlación al mismo directorio que el diseño de origen). Para obtener información acerca de la agrupación de campos de diseño de correlación, consulte el tema relacionado.

Como alternativa, puede abrir el archivo de correlación utilizado actualmente para la representación, editarlo y luego reconstruirlo para el uso. Para abrir el archivo de correlación, pulse dentro de la representación del archivo de correlación con el botón derecho del ratón y seleccione **Abrir archivo de correlación** en el menú. Se abrirá el archivo de correlación en modalidad de edición. Cuando termine de editar la correlación, guárdela. A continuación, para utilizar la correlación modificada para la representación actual, pulse con el botón derecho del ratón dentro del panel **Representaciones** de la vista Memoria y seleccione **Reconstruir correlación**.

Nota: Si pulsa con el botón derecho del ratón uno o varios nodos correspondientes a un solo archivo de correlación y elige **Abrir archivo de correlación**, se abrirá el archivo XML de dicha correlación. Si pulsa con el botón derecho del ratón los nodos de varias correlaciones, la acción **Abrir archivo de correlación** del menú emergente abrirá un submenú que lista los archivos XML de todas las correlaciones seleccionadas. En esta lista puede elegir el archivo XML que desee abrir.

Tareas relacionadas

“Agrupar campos de diseño de correlación” en la página 372
Puede organizar los campos de los diseños de correlación en grupos para facilitar el trabajo con ellos.

Editar memoria correlacionada y descripciones de campos en la vista Memoria
Para cambiar el contenido de la memoria correlacionada o las descripciones de campos en la vista Memoria, complete los pasos siguientes.

Procedimiento

1. En el panel **Representaciones** de la vista Memoria, seleccione la representación correlacionada en la que desea realizar el cambio.
2. Desplácese hasta el campo que desea cambiar. También puede pulsar la representación con el botón derecho del ratón y elegir la opción **Buscar campo** del menú. Con ello abrirá el recuadro de diálogo **Buscar campo**, en el que puede especificar un campo al que desplazarse.
3. Realice una de las tareas siguientes para cambiar el contenido de la memoria:
   a) Efectúe una doble pulsación en el campo o su valor. Esto situará el valor del campo en modalidad de edición y podrá especificar un valor válido para esa ubicación de memoria.
   b) Pulse el campo o su valor con el botón derecho del ratón y seleccione **Editar valor** en el menú. Esto situará el valor del campo en modalidad de edición y podrá especificar un valor válido para esa ubicación de memoria.
4. Realice una de las tareas siguientes para cambiar una descripción de campo:
   a) Efectúe una doble pulsación en la celda **Descripción** del campo. Esto situará la descripción en modalidad de edición y podrá especificar una descripción o editar la existente.
   b) Pulse el campo con el botón derecho del ratón y elija **Editar descripción** en el menú. Esto situará la descripción en modalidad de edición y podrá especificar una descripción o editar la existente.
5. Pulse **Intro** para someter el cambio. Si realiza cambios en el contenido de la memoria, el depurador comprobará la validez del valor.

Resultados

Para editar las descripciones de varios campos a la vez, seleccione los campos utilizando las teclas Control o Mayús del teclado y, a continuación, pulse con el botón derecho del ratón y seleccione **Editar descripción** en el menú. De este modo se abre el recuadro de diálogo Editar descripción, que le permite editar las descripciones de los campos y aplicar una descripción a todos los campos seleccionados.

Nota: Solamente puede editar descripciones de campos. No puede editar las descripciones de elementos particionados o grupos de organización.

Eliminar memoria correlacionada de la vista Memoria
Para eliminar la correlación de memoria actual del panel **Representaciones** de la vista memoria, pulse el botón **Eliminar representación** ().

Agrupar campos de diseño de correlación
Puede organizar los campos de los diseños de correlación en grupos para facilitar el trabajo con ellos.

Procedimiento

1. Pulse con el botón derecho del ratón dentro del panel **Representaciones** de la vista Memoria y pulse **Gestionar grupos**. Se abrirá el recuadro de diálogo Gestionar grupos. En este recuadro de diálogo puede añadir y eliminar nombres de grupo correspondientes a la correlación de memoria actual.
2. Una vez añadidos los grupos de memoria deseados, puede añadirles campos de diseño de correlación:
   a) Seleccione el campo o campos que desea añadir a un grupo. Para seleccionar varios campos, utilice las teclas Control o Mayús.
   b) Pulse la selección con el botón derecho del ratón y seleccione **Establecer grupos** en el menú emergente. Esta opción de menú abrirá un subgrupo en el que puede elegir el grupo al que desea añadir el campo o añadir el campo a todos los grupos.
3. Después de establecer el grupo o grupos con los que desea trabajar, puede utilizar estos valores de grupo para filtrar campos del diseño de correlación para facilitar la visualización. Para ello, pulse con

el botón derecho del ratón dentro del panel y seleccione **Mostrar grupo** en el menú emergente. Esta opción de menú abrirá un subgrupo en el que puede elegir el grupo que desea visualizar. Cuando seleccione el grupo, éste filtrará los campos que no pertenezcan al grupo. Si desea que el panel visualice todos los campos, asegúrese de seleccionar **Mostrar correlación entera**.

4. Después de añadir grupos a la correlación visualizada actualmente, puede exportar los cambios realizados. Para ello, pulse con el botón derecho del ratón dentro del panel **Representaciones** y seleccione **Exportar archivo de correlación** en el menú emergente. Si existen errores en el archivo de correlación, se le indicará mediante un diálogo de error y no podrá exportar la correlación. Si no existen errores en el archivo de correlación, se le solicitará que busque la ubicación en la que desea guardar la correlación. Si guarda la correlación en la ubicación en la que reside la correlación original, la correlación exportada sobrescribirá la correlación original.

Atención: La información de grupos no se guarda en un diseño de correlación de memoria a menos que se seleccione la preferencia **Al editar grupos y descripciones, guardar siempre los cambios en el archivo** de Correlación de memoria o que exporte explícitamente la información a un archivo. De lo contrario, si ha añadido información de grupos a una representación y, a continuación, elimina la representación sin exportar el archivo, la información de grupos se descartará.

Resultados

Si efectúa cambios de grupo en una correlación y desea descartarlos todos, pulse con el botón derecho del ratón dentro del panel **Representaciones** y seleccione **Reconstruir correlación**. Se abrirá un diálogo que le solicitará si desea conservar la información de grupos no guardada al reconstruir la correlación. Si pulsa **No**, todos los cambios efectuados en los grupos de correlación se descartarán. Para conservar la información de grupos, pulse **Sí**.

Buscar y expandir campos

Al trabajar con correlaciones de memoria, hay acciones disponibles como ayuda para la localización de campos.

Acerca de esta tarea

De forma predeterminada, al representar memoria con una correlación, sólo se expande el elemento raíz. Todos los demás elementos quedan contraídos. Para expandir todos los campos (excepto los tipos de correlaciones) de una correlación, pulse la representación de correlación con el botón derecho del ratón y seleccione **Expandir toda la correlación** en el menú emergente. Para expandir y visualizar todos los hijos de un nodo individual, pulse con el botón derecho del ratón y seleccione **Expandir <node name>**, donde **<node name>** es el nodo que ha seleccionado para expandir. Si selecciona esta acción, los tipos de correlaciones dentro del nodo no se expandirán.

Para abrir el recuadro de diálogo Buscar campo, pulse la representación de correlación con el botón derecho del ratón y seleccione **Buscar campo** en el menú emergente. Este recuadro de diálogo permite especificar un campo al que desea desplazarse. Puede realizar búsquedas por campo, descripción, vía de acceso o grupo, seleccionando el filtro de búsqueda en el recuadro de selección desplegable **Elegir un filtro de búsqueda**. A continuación, especifique la serie por la que desea buscar, en el campo **Entre una serie de búsqueda**. Por ejemplo, si desea buscar campos de un grupo llamado `GrupoA`, seleccione el filtro de búsqueda **Grupo** y escriba `GrupoA` en el campo de serie de búsqueda. Esto hará que en la tabla del recuadro de diálogo solamente se muestren los campos de dicho grupo; en esta tabla puede seleccionar el campo que desea encontrar.

Antes de expandir los nodos y correlaciones, aún no están *creados* en la vista Memoria. Si desea buscar un campo, el recuadro de diálogo Buscar campo sólo se llena con los campos que se han creado. Para visualizar elementos largos que se hayan dividido en la vista Memoria (a efectos de visualización), marque el recuadro de selección **Mostrar elementos particionados**. Para construir toda la correlación de forma que todos los elementos (excepto los de la correlación de tipos) se visualicen en el recuadro de lista, marque el recuadro de selección **Construir toda la correlación**. De forma predeterminada, este recuadro de selección está seleccionado. Este valor (construir la correlación antes de abrir el recuadro de diálogo

Buscar) también puede establecerse en las preferencias de correlación de memoria. Para obtener más información acerca de este valor, consulte el tema relacionado.

Cuando se abre el recuadro de diálogo Buscar campo, todos los campos creados se visualizan en el recuadro de lista. Puede controlar las columnas que se muestran en este recuadro de lista pulsando **Elegir columnas**. Si sólo desea realizar búsquedas en un campo, seleccione el filtro de búsqueda **Campo** y especifique el nombre de campo (o parte del nombre de campo) que desee encontrar en el campo de serie de búsqueda. A medida que escriba, el contenido del recuadro de lista de reducirá para incluir sólo los campos que empiezan por la entrada especificada en el campo. Esto ayuda a especificar el nombre del campo que desea buscar. En el campo, puede especificar filtros para los campos (incluido el uso del comodín '*' para representar cero o más caracteres o el comodín '?' para representar cualquier carácter). También se utilizan comodines de la misma forma que cuando se realizan búsquedas mediante otros filtros.

Cuando especifique un campo en el recuadro de diálogo Buscar campo y pulse **Aceptar**, la representación de correlación de memoria resaltará el campo se lo encuentra en la correlación.

Añadir varias correlaciones de memoria
Al añadir una expresión, variable o registro a la vista Memoria, puede hacerlo para varias correlaciones.

Acerca de esta tarea

Puede añadir las correlaciones de una en una o, al seleccionar una correlación, puede utilizar las teclas Mayús o Control para seleccionar varias correlaciones. Si lo hace así, se creará una representación de correlación de memoria para cada correlación seleccionada. Las representaciones estarán separadas por pestañas.

Depuración de una transacción CICS local con TXSeries o CICS TX

Puede depurar una transacción CICS local con TXSeries o CICS TX.

Acerca de esta tarea

Para configurar IBM Debug for Linux en x86 con TXSeries o CICS TX, siga estos pasos:

Procedimiento

1. Cambie el atributo AllowDebugging de la definición de región (RD) de la región CICS a sí. Puede utilizar el mandato siguiente para cambiar la configuración de la región para la depuración:

   ```
   cicsupdate -r REGION_NAME -c rd AllowDebugging=yes
   ```

2. Compile el programa CICS COBOL que se va a depurar con el distintivo **-a** si utiliza **cicstcl**, o añadiendo la opción **-g** si está compilando utilizando el mandato de compilador **cob2** . Por ejemplo:

   ```
   cicstcl -a -lIBMCOB prog.ccp
   ```

   Nota: Las transacciones que empiezan por la letra C no se pueden depurar, porque están reservadas para uso interno de CICS .

3. Establezca la siguiente variable de entorno en el archivo de entorno de la región e inicie en frío la región:

   ```
   DER_DBG_PATH=Path_to_source_files [To inform IDEBUG to pick the source file
                                      from the specified path]
   CICS_IDEBUG_LIBPATH=/opt/ibm/cobol/debug/usr/lib/
   ```

4. Configure la región para utilizar el depurador distribuido a través de la transacción proporcionada por CDCN:

   a. Conéctese a la región utilizando el cliente cicsterm (o podría utilizar cualquier otro emulador de terminal basado en 3270).

b. Ejecute la transacción CDCN. La primera pantalla de la transacción CDCN se muestra a continuación:

```
 CDCN              CICS Debugging Configuration Transaction

        DISPLAY :(         )                                DEBUG  : ON

 To configure for a ter
minal specify the TERMID            TERMID : (    )

 To configure for a system specify the SYSID               SYSID  : (    )

 To configure for a transaction specify the TRANSID        TRANSID: (    )

 To configure for a program specify the PROGRAM            PROGRAM: (    )


 ENTER:    COMMIT SELECTION
 PF1 : HELP                PF2 : DEBUG ON/OFF        PF3 : EXIT
 PF4 : MESSAGES            PF5 : UNDEFINED           PF6 : UNDEFINED
 PF7 : UNDEFINED           PF8 : UNDEFINED           PF9 : UNDEFINED
 PF10: UNDEFINED           PF11: UNDEFINED           PF12: UNDEFINED
```

c. Utilice CDCN para configurar los recursos de CICS® adecuados como, por ejemplo, una transacción específica o un programa específico. CDCN también le permite depurar todos los programas que se ejecutan en un terminal especificado o que se direccionan a un sistema específico. Si especifica más de un recurso, CDCN tiene el siguiente orden de prioridad:

   i) IdTerm

   ii) IdSist

   iii) IdTran

   iv) Program

   Por ejemplo, para depurar un recurso PROGRAM denominado CUSTECIC, realice los pasos siguientes:

   i) En la pantalla CDCN, establezca el campo DISPLAY: en la dirección IP de la máquina y el puerto donde se ejecuta la interfaz de usuario de Distributed Debugger.

   ii) Para depurar solo el programa CUSTECIC, establezca el campo PROGRAM en CUSTECIC. La figura siguiente muestra el contenido de la pantalla CDCN después de los cambios:

```
CDCN              CICS Debugging Configuration Transaction

       DISPLAY : 9.100.194.80:9005                        DEBUG  : ON

To configure for a terminal specify the TERMID            TERMID : (    )

To configure for a system specify the SYSID               SYSID  : (    )

To configure for a transaction specify the TRANSID        TRANSID: (    )

To configure for a program specify the PROGRAM            PROGRAM: CUSTECIC


ENTER:    COMMIT SELECTION
PF1 : HELP                PF2 : DEBUG ON/OFF        PF3 : EXIT
PF4 : MESSAGES            PF5 : UNDEFINED           PF6 : UNDEFINED
PF7 : UNDEFINED           PF8 : UNDEFINED           PF9 : UNDEFINED
PF10: UNDEFINED           PF11: UNDEFINED           PF12: UNDEFINED
```

iii) Pulse Intro. Se visualizan los mensajes siguientes para mostrar que el depurador se ha configurado correctamente:

```
There are 2 messages:

ERZI04066I: Successfully configured debugging on program 'CUSTECIC'
ERZI04072I: The display to be used for the debugging information is
'9.100.194.80:9005 '
```

iv) Pulse Intro, seguido de F3, para salir de la transacción CDCN.

Qué hacer a continuación

Para iniciar la depuración de programas CICS utilizando el Eclipse IDE en la estación de trabajo, siga estos pasos:

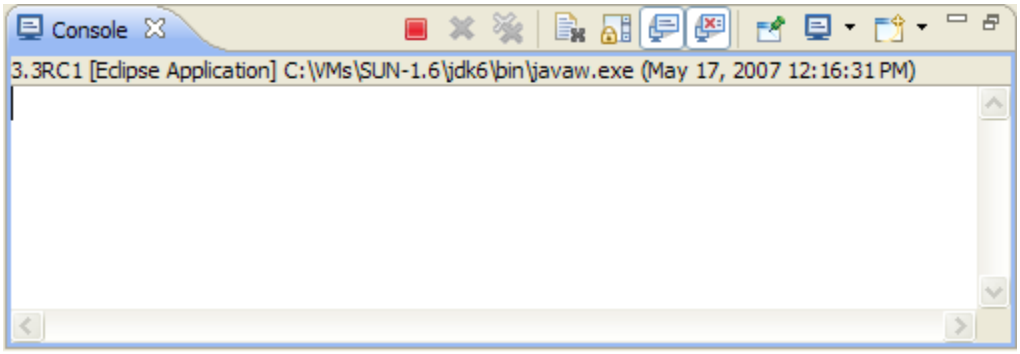
1. Abra la perspectiva de depuración Eclipse IDE y pulse para empezar a escuchar en el puerto configurado para la depuración desde la región CICS utilizando la transacción CDCN. El puerto predeterminado es 8001.
2. Ejecute la transacción o programa y asegúrese de que la transacción o programa ya está configurado para la depuración utilizando la transacción CDCN.
3. Ahora puede ver el origen del programa visualizado en el Eclipse IDE y el programa está bajo el control del depurador. Puede utilizar las opciones del depurador para continuar depurando el programa CICS COBOL.

referencias

Esta sección proporciona información de referencia sobre vistas en IBM Debug for Linux en x86.

Vista de consola

La vista Consola muestra una variedad de tipos de consola en función del tipo de desarrollo y del conjunto actual de valores de usuario.



Las tres consolas que se proporcionan de forma predeterminada con Eclipse Platform son las siguientes:

- La consola de procesos
- Consola de rastreo de pila
- La consola CVS

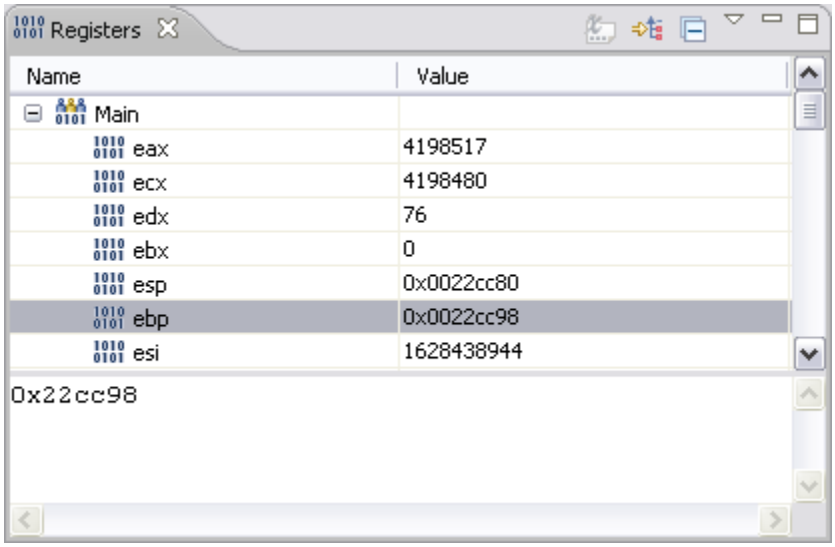
Puede cambiar los valores de las consolas seleccionando Ejecutar/Depurar > Consola en la página de preferencias de la consola.

Los mandatos disponibles en la vista Consola se listan a continuación.

Tabla 34. Mandatos de vista de consola			
Mandato	**Name**	**Descripción**	**Disponibilidad**
	Borrar consola	Borra la consola activa actualmente y está disponible como un mandato de vista y un elemento de menú contextual.	Menú contextual y acción de vista
	Mostrar consola seleccionada	Abre un listado de consolas actuales y le permite seleccionar cuál desea ver.	Ver acción
	Abrir consola	Abre una nueva consola del tipo seleccionado.	Ver acción
	Clavija	Pines la consola actual para permanecer en la parte superior de todas las demás consolas.	Ver acción
	Bloqueo de desplazamiento	Cambia si el bloqueo de desplazamiento debe estar habilitado o no en la consola actual.	Menú contextual y acción de vista

Vista de registros

La vista Registros de la perspectiva Depurar lista información sobre los registros en un marco de pila seleccionado. Los valores que han cambiado se resaltan en la vista Registros cuando se detiene el programa.



Registra opciones de barra de herramientas de vista

La tabla siguiente lista los iconos que se muestran en la barra de herramientas de la vista Registros.

Icono	**Name**	**Descripción**
	Mostrar nombres de tipo	Muestra el tipo (por ejemplo, **int**) junto a cada valor de registro.

Icono	Name	Descripción
	Mostrar estructura lógica	Cambia si las estructuras lógicas deben mostrarse en la vista o no.
	Contraer todo	Contrae todos los registros expandidos actualmente.
	Menú Ver > Diseño	Proporciona varias opciones de diseño para la vista Registros.

Registra mandatos de menú contextual de vista

Los mandatos de menú contextual de la vista Registros incluyen:

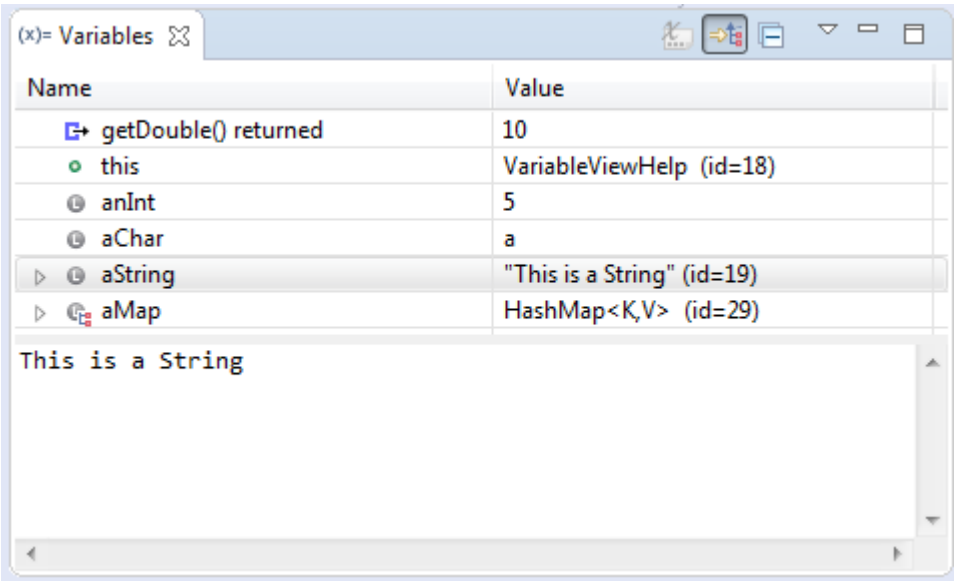
Icono	Name	Descripción
	Añadir grupo de registro	Abre el diálogo **Registrar grupo** que le permite definir un grupo de registro que se muestra en la vista Registros.
	Asignar valor	Asigna un valor al registro seleccionado.
	Convertir a tipo ...	Abre el diálogo **Convertir a tipo** .
	Cambiar valor ...	Abre el diálogo **Establecer valor** para cambiar el valor de los registros seleccionados.
	Asistencia de contenido	Abre un diálogo de asistencia de contenido en la posición actual del cursor.
	Copiar	Copia el texto o elementos seleccionados actualmente en el portapapeles.
	Copiar registros	Copia los nombres de registro y el contenido en el portapapeles.
	Crear expresión de observación	Convierte el registro seleccionado en una expresión de observación.
	Cortar	Copia el texto o elemento seleccionado actualmente en el portapapeles y elimina el elemento.
	disable	Inhabilita el registro seleccionado.
	Mostrar como matriz ...	Abre el diálogo **Mostrar como matriz** que le permite especificar el inicio y la longitud de la matriz.
	Editar grupo de registro	Abre el diálogo **Registrar grupo** para editar el grupo de registro seleccionado.

Icono	Name	Descripción
	Habilitar	Habilita el registro seleccionado.
	Buscar ...	Abre el diálogo **Buscar** que le permite buscar elementos específicos en la vista.
	Buscar/Sustituir	Abre el diálogo **Buscar/ Sustituir** .
	Formato	Selecciona un tipo de formato. Las opciones incluyen Predeterminado, Decimal, Hexadecimal, Octal y Binario.
	Longitud máxima ...	Abre el diálogo **Configurar panel de detalles** para establecer el número máximo de caracteres a visualizar. El valor predeterminado es 10000.
	Pegar	Pega el contenido del portapapeles actual como texto.
	Eliminar grupo de registros	Elimina el grupo de registro seleccionado actualmente.
	Restaurar grupos de registros predeterminados	Restaura los grupos de registro originales.
	Restaurar tipo original	Devuelve el registro seleccionado al tipo original.
	Seleccionar todo	Selecciona todo el contenido del editor.
	Ajustar texto	Se activa para ajustar el contenido del texto dentro del área visible del panel Detalles de la vista Registros.

Vista Variables

La vista Variables muestra información sobre las variables asociadas con el marco de pila seleccionado en la vista Depurar. Al depurar un programa Java™ , se pueden seleccionar variables para que se visualice información más detallada en el panel de detalles. Además, los objetos Java se pueden expandir para mostrar los campos que contiene la variable.

La vista Variables se muestra con columnas. El panel de detalles es el área en la parte inferior de la vista para mostrar texto.



Hay muchos mandatos disponibles en la vista Variables:

- Los mandatos de visualización de vista afectan a las variables que se muestran y cómo se presentan.
- El panel de detalles tiene muchos mandatos disponibles pulsando con el botón derecho del ratón en él.
- Los mandatos de diseño de vista afectan al modo en que se orienta el panel de detalles y a si se visualizan las columnas.
- Otros mandatos se listan a continuación:

Tabla 35. Mandatos de la vista Variables

Mandato	Name	Descripción	Disponibilidad
	Todas las instancias	Abre un diálogo emergente que muestra una lista de todas las instancias del tipo Java seleccionado. La máquina virtual Java debe dar soporte a la recuperación de instancias.	Menú contextual
	Todas las referencias	Abre un diálogo emergente que muestra una lista de todos los objetos Java que tienen referencias a la variable seleccionada. La máquina virtual Java debe dar soporte a la recuperación de referencia.	Menú contextual
	Cambiar valor ...	Permite cambiar el valor de la variable seleccionada subyacente.	Menú contextual

Tabla 35. Mandatos de la vista Variables (continuación)

Mandato	Name	Descripción	Disponibilidad
	Contraer todo	Contrae todas las variables expandidas actualmente.	Ver acción
	Copiar variables	Copia las variables seleccionadas en el portapapeles del sistema.	Menú contextual
	Crear expresión de observación	Permite crear una expresión de observación para la variable seleccionada.	Menú contextual
	Buscar ...	Abre el diálogo de búsqueda para buscar elementos en la vista Variables.	Menú contextual
	Inspeccionar	Crea una nueva sentencia de inspección para la variable seleccionada y la añade a la vista de expresiones.	Menú contextual
	Puntos de interrupción de instancia ...	Permite filtrar puntos de interrupción existentes en la instancia de variable seleccionada.	Menú contextual
	Preferencias de Java ...	Abre varias páginas de preferencias que contienen opciones que afectan a la vista.	Ver acción
	Nuevo formateador de detalles ...	Le permite crear su propio formateador de detalles para ese tipo de variable.	Menú contextual
	Abrir tipo real	Abre el tipo real de la variable seleccionada.	Menú contextual
	Abrir jerarquía de tipos reales	Abre la jerarquía de tipos real para el tipo real de la variable seleccionada.	Menú contextual
	Abrir tipo declarado	Abre el tipo declarado para la variable seleccionada en un editor nuevo.	Menú contextual

Tabla 35. Mandatos de la vista Variables (continuación)			
Mandato	**Name**	**Descripción**	**Disponibilidad**
	Abrir jerarquía de tipos declarada	Abre la jerarquía de tipos para el tipo declarado de la variable seleccionada.	Menú contextual
	Seleccionar todo	Selecciona todas las variables de la vista.	Menú contextual
	Mostrar estructura lógica	Permite seleccionar un formateador para mostrar la variable de tipo de estructura lógica seleccionada.	Menú contextual
	Editar estructura lógica	Abre la página de preferencias para editar estructuras lógicas.	Menú de subcontexto
	Mostrar detalles como ...	Le permite seleccionar un panel de detalles diferente para mostrar información detallada sobre las variables seleccionadas.	Menú contextual
	Conmutar punto de observación	Crea un nuevo punto de observación en el campo seleccionado actualmente o elimina el punto de observación si ya existe uno.	Menú contextual

Obtención de listados

Obtenga la información que necesita para depurar solicitando el listado de compilador adecuado con el uso de opciones de compilador.

Acerca de esta tarea

Atención: Los listados producidos por el compilador no son una interfaz de programación y están sujetos a cambios.

Tabla 36. ***Utilización de opciones de compilador para obtener listados***

Utilizar	Listado	Contenido	Opción de compilador
Para comprobar una lista de las opciones en vigor para el programa, las estadísticas sobre el contenido del programa y los mensajes de diagnóstico sobre la compilación Para comprobar el entorno local en vigor durante la compilación	Listado corto	• Lista de opciones en vigor para el programa • Estadísticas sobre el contenido del programa • Mensajes de diagnóstico sobre la compilación[1] Línea de entorno local que muestra el entorno local en vigor	NOSOURCE, NOXREF, NOVBREF, NOMAP, NOLIST
Para ayudar a probar y depurar el programa; para tener un registro después de que se haya depurado el programa	Listado de origen	Copia del origen	"fuente" en la página 297
Para encontrar determinados elementos de datos; para ver la asignación de almacenamiento final después de que se haya contabilizado la reentrada o la optimización; para ver dónde se definen los programas y comprobar sus atributos	Mapa de elementos DATA DIVISION	Todos los elementos DATA DIVISION y todos los elementos declarados implícitamente Resumen de mapa incorporado (en el margen derecho del listado para líneas del DATA DIVISION que contienen declaraciones de datos) Correlación de programas anidados (si el programa contiene programas anidados)	"MAP" en la página 288[2]
Para buscar dónde se define, se hace referencia o se modifica un nombre; para determinar el contexto (por ejemplo, si se ha utilizado una sentencia en un bloque PERFORM) en el que se hace referencia a un procedimiento; para determinar el conjunto de datos del que se ha obtenido un libro de copias	Listado de referencias cruzadas ordenadas de nombres; listado de referencias cruzadas ordenadas de sentencias COPY/BASIS y conjuntos de datos	Nombres de datos, nombres de procedimiento y nombres de programa; referencias a estos nombres COPY/BASIS nombres de texto y nombres de biblioteca, y los conjuntos de datos de los que se han obtenido los libros de copias asociados Las referencias cruzadas modificadas incorporadas proporcionan números de línea en los que se han definido nombres de datos y nombres de procedimiento	"XREF" en la página 306[2,3]

*Tabla 36. **Utilización de opciones de compilador para obtener listados** (continuación)*			
Utilizar	**Listado**	**Contenido**	**Opción de compilador**
Para encontrar la sentencia anómala en un programa o la dirección en almacenamiento de un elemento de datos que se mueve mientras el programa se está ejecutando	Código PROCEDURE DIVISION y código de ensamblador producidos por el compilador[3]	Código generado	"lista" en la página 287[2,4]
Para buscar una instancia de una determinada sentencia	Listado alfabético de sentencias	Cada sentencia utilizada, número de veces que se ha utilizado cada sentencia, números de línea donde se ha utilizado cada sentencia	"VBREF" en la página 305

1. Para eliminar mensajes, desactive las opciones (como FLAG) que rigen el nivel de información de diagnóstico de compilación.
2. Para utilizar los números de línea en el programa compilado, utilice la opción de compilador NUMBER . El compilador comprueba la secuencia de los números de línea de sentencia fuente en las columnas 1 a 6 a medida que se leen las sentencias. Cuando encuentra un número de línea fuera de secuencia, el compilador le asigna un número con un valor uno superior al número de línea de la sentencia anterior. El nuevo valor se marca con dos asteriscos. Un mensaje de diagnóstico que indica un error fuera de secuencia se incluye en el listado de compilación.
3. El contexto de la referencia de procedimiento se indica mediante los caracteres que preceden al número de línea.
4. El listado del ensamblador se graba en el archivo de listado (un archivo que tiene el mismo nombre que el programa fuente pero con el sufijo .wlist).

"Ejemplo: listado corto" en la página 384
"Ejemplo: salida SOURCE y NUMBER" en la página 386
"Ejemplo: salida MAP" en la página 387
"Ejemplo: resumen de mapa incorporado" en la página 388
"Ejemplo: correlación de programa anidado" en la página 390
"Ejemplo: Salida XREF: referencias cruzadas de nombre de datos" en la página 391
"Ejemplo: Salida XREF: referencias cruzadas de nombre de programa" en la página 392
"Ejemplo: Salida XREF: COPY/BASE referencias cruzadas" en la página 392
"Ejemplo: Salida XREF: referencia cruzada incorporada" en la página 393
"Ejemplo: salida del compilador VBREF" en la página 394

Tareas relacionadas

"Generación de una lista de mensajes del compilador" en la página 246

Referencias relacionadas

"Mensajes y listados de errores detectados por el compilador" en la página 246

Ejemplo: listado corto

Los números entre paréntesis que se muestran en la lista siguiente corresponden a las explicaciones numeradas que siguen a la lista. Con fines ilustrativos, se han introducido deliberadamente algunos errores que causan mensajes de diagnóstico.

```
PROCESS(CBL) statements:   (1)
```

```
 CBL    NOSOURCE,NOXREF,NOVBREF,NOMAP,NOLIST    (2)
Options in effect:    (3)
 NOADATA
   ADDR(64)
   QUOTE
   ARITH(COMPAT)
   BINARY(NATIVE)
   CALLINT(NODESCRIPTOR)
   CHAR(NATIVE)
 NOCICS
   COLLSEQ(BINARY)
 NOCOMPILE(S)
 NOCURRENCY
   DATETIME(1900,40)
 NODATEPROC
 NODIAGTRUNC
 NODYNAM
 NOEXIT
   FLAG(I,I)
 NOFLAGSTD
   FLOAT(NATIVE)
   LINECOUNT(60)
 NOLIST
   LSTFILE(LOCALE)
 NOMAP
 NOMDECK
   NCOLLSEQ(BINARY)
   NSYMBOL(NATIONAL)
 NONUMBER
 NOOPTIMIZE
   PGMNAME(LONGUPPER)
   SEPOBJ
   SEQUENCE
 NOSOSI
 NOSOURCE
   SPACE(1)
   SPILL(512)
 NOSQL
   SRCFORMAT(COMPAT)
 NOSSRANGE
   TERM
 NOTEST
 NOTHREAD
   TRUNC(STD)
   UTF16(NATIVE)
 NOVBREF
 NOWSCLEAR
 NOXREF
   YEARWINDOW(1900)
   ZWB
```

```
LineID  Message code  Message text    (4)
        IGYDS0139-W   Diagnostic messages were issued during processing of compiler options.  These messages are
                     located at the beginning of the listing.
   193  IGYDS1050-E   File "LOCATION-FILE" contained no data record descriptions.  The file definition was discarded.
   889  IGYPS2052-S   An error was found in the definition of file "LOCATION-FILE".  The reference to this file
                     was discarded.
                       Same message on line:    983
   993  IGYPS2121-S   "WS-DATE" was not defined as a data-name.  The statement was discarded.
                       Same message on line:    994
   995  IGYPS2121-S   "WS-TIME" was not defined as a data-name.  The statement was discarded.
                       Same message on line:    996
   997  IGYPS2053-S   An error was found in the definition of file "LOCATION-FILE".  This input/output statement
                     was discarded.
                       Same message on line:   1009
  1008  IGYPS2121-S   "LOC-CODE" was not defined as a data-name.  The statement was discarded.
  1219  IGYPS2121-S   "COMMUTER-SHIFT" was not defined as a data-name.  The statement was discarded.
                       Same message on line:   1240
  1220  IGYPS2121-S   "COMMUTER-HOME-CODE" was not defined as a data-name.  The statement was discarded.
                       Same message on line:   1241
  1222  IGYPS2121-S   "COMMUTER-NAME" was not defined as a data-name.  The statement was discarded.
                       Same message on line:   1243
  1223  IGYPS2121-S   "COMMUTER-INITIALS" was not defined as a data-name.  The statement was discarded.
                       Same message on line:   1244
  1233  IGYPS2121-S   "WS-NUMERIC-DATE" was not defined as a data-name.  The statement was discarded.
Messages    Total    Informational    Warning    Error   Severe   Terminating    (5)
Printed:       21                        2        1       18
  * Statistics for COBOL program SLISTING:    (6)
* Source records = 1765
* Data Division statements = 277
* Procedure Division statements = 513
Locale = en_US.ISO8859-1                    (7)
End of compilation 1,  program SLISTING,  highest severity: Severe.   (8)
Return code 12
```

(1)
Mensaje sobre las opciones especificadas en una sentencia PROCESS (o CBL). Este mensaje no aparece si no se ha especificado ninguna opción.

(2)
Opciones codificadas en la sentencia PROCESS (o CBL).

(3)
Estado de las opciones al inicio de esta compilación.

(4)
Diagnósticos de programa. El primer mensaje le remite a los diagnósticos de fase de biblioteca, si los hubiera. Los diagnósticos para la fase de biblioteca siempre se presentan al principio del listado.

(5)
Recuento de mensajes de diagnóstico en este programa, agrupados por nivel de gravedad.

(6)
Estadísticas de programa para el programa SLISTING.

(7)
El entorno local que ha utilizado el compilador.

(8)
Estadísticas de programa para la unidad de compilación. Cuando realiza una compilación por lotes (varios programas COBOL más externos en una sola compilación), el código de retorno es el nivel de gravedad de mensaje más alto para toda la compilación.

Ejemplo: salida SOURCE y NUMBER

En la parte del listado que se muestra a continuación, el programador numeró dos de las sentencias fuera de secuencia. Los números de nota en el listado corresponden a explicaciones numeradas que siguen al listado.

```
                    (1)
 LineID  PL SL  ----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7-|--+----8  Cross-
Reference
   (2)    (3)    (4)                                                                              |
                 087000/*****************************************************************         |
                 087100***                 D O   M A I N   L O G I C                   **         |
                 087200***                                                             **         |
                 087300*** Initialization. Read and process update transactions until **         |
                 087400*** EOE. Close files and stop run.                              **         |
                 087500*****************************************************************         |
                 087600 procedure division.                                                       |
                 087700   000-do-main-logic.                                                      |
                 087800     display "PROGRAM SRCOUT - Beginning"                                  |
                 087900     perform 050-create-stl-main-file.                                     |
                 088150     display "perform 050-create-stl-main-file finished".                  |
 088151**        088125     perform 100-initialize-paragraph                                      |
                 088200     display "perform 100-initialize-paragraph finished"                   |
                 088300     read update-transaction-file into ws-transaction-record               |
                 088400         at end                                                            |
              1  088500       set transaction-eof to true                                         |
                 088600     end-read                                                              |
                 088700     display "READ completed"                                              |
                 088800     perform until transaction-eof                                         |
              1  088900       display "inside perform until loop"                                 |
              1  089000       perform 200-edit-update-transaction                                 |
              1  089100       display "After perform 200-edit   "                                 |
              1  089200       if no-errors                                                        |
              2  089300         perform 300-update-commuter-record                                |
              2  089400         display "After perform 300-update "                               |
              1  089650       else                                                                |
 089651**     2  089600         perform 400-print-transaction-errors                              |
              2  089700         display "After perform 400-errors "                               |
              1  089800       end-if                                                              |
              1  089900       perform 410-re-initialize-fields                                    |
              1  090000       display "After perform 410-reinitialize"                            |
              1  090100       read update-transaction-file into ws-transaction-record             |
              1  090200           at end                                                          |
              2  090300         set transaction-eof to true                                       |
              1  090400       end-read                                                            |
              1  090500       display "After '2nd READ'   "                                       |
                 090600     end-perform                                                           |
```

(1)
Etiquetas de línea de escala Área A, Área B y números de columna de código fuente

(2)
Número de línea de código fuente asignado por el compilador

(3)
Nivel de anidamiento de programa (PL) y sentencia (SL)

(4)
Columnas 1 a 6 del programa (el área de número de secuencia)

Ejemplo: salida MAP

El ejemplo siguiente muestra la salida de la opción MAP . Los números utilizados en la explicación siguiente corresponden a los números que anotan la salida.

```
 Data Division Map
 (1)
 Data Definition Attribute codes (rightmost column) have the following meanings:
     D = Object of OCCURS DEPENDING    G = GLOBAL                           LSEQ= ORGANIZATION LINE
SEQUENTIAL
     E = EXTERNAL                      O = Has OCCURS clause                SEQ= ORGANIZATION SEQUENTIAL
     VLO=Variably Located Origin       OG= Group has own length definition  INDX= ORGANIZATION INDEXED
     VL= Variably Located              R = REDEFINES                        REL= ORGANIZATION RELATIVE

 (2)      (3) (4)                                          (5)      (6)         (7)           (8)
 Source   Hierarchy and                                                                     Data Def
 LineID   Data Name                                     Length(Displacement) Data Type      Attributes
      4  PROGRAM-ID IGYTCARA----------------------------------------------------------------------------------
*
    180   FD COMMUTER-FILE                                                    File            INDX
    182   1  COMMUTER-RECORD                                 80               Group
    183     2  COMMUTER-KEY                                 16(0000000)      Display
    184     2  FILLER                                       64(0000016)      Display
    186   FD COMMUTER-FILE-MST                                                File            INDX
    188   1  COMMUTER-RECORD-MST                           80                 Group
    189     2  COMMUTER-KEY-MST                           16(0000000)        Display
    190     2  FILLER                                     64(0000016)        Display
    192   FD LOCATION-FILE                                                    File            SEQ
    203   FD UPDATE-TRANSACTION-FILE                                          File            SEQ
    208   1  UPDATE-TRANSACTION-RECORD                     80                 Display
    216   FD PRINT-FILE                                                       File            SEQ
    221   1  PRINT-RECORD                                 121                Display
    228   1  WORKING-STORAGE-FOR-IGYCARA                    1                Display
```

(1)
Explicaciones de los códigos de atributo de definición de datos.

(2)
Número de línea de origen donde se ha definido el elemento de datos.

(3)
Número o definición de nivel. El compilador genera este número de la forma siguiente:

- El primer nivel de cualquier jerarquía es siempre 01. Aumente 1 para cada nivel (cualquier elemento que haya codificado como de nivel 02 a 49).
- Los números de nivel 66, 77 y 88, y los indicadores FD y SD, no se modifican.

(4)
Nombre de datos que se utiliza en el módulo de origen en orden de origen.

(5)
Longitud del elemento de datos. Valor de localizador base.

(6)
Desplazamiento hexadecimal desde el principio de la estructura contenedora.

(7)
Tipo de datos y uso.

(8)
Códigos de atributo de definición de datos. Las definiciones se explican en la parte superior del mapa de `DATA DIVISION`.

"Ejemplo: resumen de mapa incorporado" en la página 388
"Ejemplo: correlación de programa anidado" en la página 390

Referencias relacionadas
"Términos y símbolos utilizados en la salida de MAP" en la página 389

Ejemplo: resumen de mapa incorporado

El ejemplo siguiente muestra un resumen de mapa incorporado a partir de la especificación de la opción `MAP`. El resumen aparece en el margen derecho del listado para las líneas del `DATA DIVISION` que contienen declaraciones de datos.

```
 000002                Identification Division.                                  |
 000003                                                                          |
 000004                Program-id.    EMBMAP.                                    |
    . . .                                                                        |
 000176                Data division.                                            |
 000177                File section.                                             |
 000178                                                                          |
 000179                                                                          |
 000180                FD  COMMUTER-FILE                                         |
 000181                    record 80 characters.                                 |      (1)   (2)
 000182                  01 commuter-record.                                     |       80
 000183                    05 commuter-key                PIC x(16).             |
16(0000000)
 000184                    05 filler                      PIC x(64).             |
64(0000016)
    . . .                                                                        |
 000221        IA1620    01 print-record                 pic x(121).            |      121
    . . .                                                                        |
 000227                Working-storage section.                                  |
 000228                  01 Working-storage-for-EMBMAP     pic x.                |        1
 000229                                                                          |
 000230                  77 comp-code                     pic S9999 comp.       |        2
 000231                  77 ws-type                       pic x(3)   value spaces.|       3
 000232                                                                          |
 000233                                                                          |
 000234                  01 i-f-status-area.                                     |        2
 000235                    05 i-f-file-status             pic x(2).             |
2(0000000)
 000236                      88 i-o-successful            value zeroes.         |IMP
 000237                                                                          |
 000238                                                                          |
 000239                  01 status-area.                                         |        8
 000240                    05 commuter-file-status       pic x(2).             |(3)
2(0000000)
 000241                      88 i-o-okay                  value zeroes.         |IMP
    . . .                                                                        |
 000246                                                                          |
 000247                  77 update-file-status            pic xx.               |        2
 000248                  77 loccode-file-status           pic xx.               |        2
 000249                  77 updprint-file-status          pic xx.               |        2
    . . .                                                                        |
 000877                procedure division.                                       |
 000878                  000-do-main-logic.                                      |
 000879                    display "PROGRAM EMBMAP - Beginning".                 |
 000880                    perform 050-create-stl-main-file.                   |931
     . . .                                                                      |
```

(1)
Longitud decimal del elemento de datos

(2)
Desplazamiento hexadecimal desde el principio del valor del localizador base

(3)
Símbolos de definición especiales:

UND
El nombre de usuario no está definido.

DUP
El nombre de usuario se define más de una vez.

IMP
Nombre definido implícitamente, como registros especiales o constantes figurativas.

IFN
Una referencia de función intrínseca.

EXT
Una referencia externa.

Términos y símbolos utilizados en la salida de MAP

La tabla siguiente describe los términos y símbolos utilizados en los listados generados por la opción de compilador MAP .

Tabla 37. ***Términos y símbolos utilizados en la salida de MAP***

TERM	Descripción
ALFABÉTICO	Alfabético (PICTURE A)
EDICIÓN ALFA	Alfabético-editado
UN-EDITAR	Alfanumérico-editado
BINARIO	Binario (USAGE BINARY, COMPUTATIONALo COMPUTATIONAL-5)
COMP-1	Coma flotante interna de precisión simple (USAGE COMPUTATIONAL-1)
COMP-2	Coma flotante interna de precisión doble (USAGE COMPUTATIONAL-2)
DBCS	DBCS (USAGE DISPLAY-1)
DBCS-EDITAR	DBCS editado
DISP-FLOAT	Visualizar coma flotante (USAGE DISPLAY)
VISUALIZAR	Alfanumérico (PICTURE X)
DISP-NUM	Decimal con zona (USAGE DISPLAY)
DISP-NUM-EDIT	Numérico-editado (USAGE DISPLAY)
FD	Definición de archivo
FUNCIÓN-PTR	Puntero a una función invocable externamente (USAGE FUNCTION-POINTER)
Grupo	Grupo de longitud fija alfanumérico
GRP-VARLEN	Grupo de longitud variable alfanumérico
ÍNDICE	Índice (USAGE INDEX)
NOMBRE ÍNDICE	Nombre de índice
NACIONAL	Categoría nacional (USAGE NATIONAL)
NAT-EDITAR	Edición nacional (USAGE NATIONAL)
NAT-FLOAT	Coma flotante nacional (USAGE NATIONAL)
GRUPO DE NAT	Grupo nacional (GROUP-USAGE NATIONAL)

*Tabla 37. **Términos y símbolos utilizados en la salida de MAP** (continuación)*	
TERM	**Descripción**
NAT-GRP-VARLEN	Grupo de longitud variable nacional (GROUP-USAGE NATIONAL)
NAT-NÚM	Decimal nacional (USAGE NATIONAL)
NAT-NUM-EDIT	Numérico nacional editado (USAGE NATIONAL)
DEC EMPAQUETADO	Decimal interno (USAGE PACKED-DECIMAL o COMPUTATIONAL-3)
PUNTERO	Puntero (USAGE POINTER)
PROCEDIMIENTO-PTR	Puntero a un programa invocable externamente (USAGE PROCEDURE-POINTER)
DE	Ordenar definición de archivo
01-49, 77	Números de nivel para descripciones de datos
66	Número de nivel para RENAMES
88	Número de nivel para nombres de condición

Ejemplo: correlación de programa anidado

Este ejemplo muestra una correlación de procedimientos anidados producidos especificando la opción de compilador MAP . Los números entre paréntesis hacen referencia a las notas que siguen el ejemplo.

```
Nested Program Map

    (1)
Program Attribute codes (rightmost column) have the following meanings:
   C = COMMON
   I = INITIAL
   U = PROCEDURE DIVISION USING...

   (2)   (3)      (4)                                      (5)
Source Nesting                                           Program
LineID Level   Program Name from PROGRAM-ID paragraph     Attributes
     2         NESTED. . . . . . . . . . . . . . . . . .
    12     1     X1. . . . . . . . . . . . . . . . . . .
    20     2       X11 . . . . . . . . . . . . . . . . .
    27     2       X12 . . . . . . . . . . . . . . . . .
    35     1     X2. . . . . . . . . . . . . . . . . . .
```

(1)
Explicaciones de los códigos de atributo de programa

(2)
Número de línea fuente donde se ha definido el programa

(3)
Profundidad de anidamiento de programa

(4)
Nombre de programa

(5)
Códigos de atributo de programa

Ejemplo: Salida XREF: referencias cruzadas de nombre de datos

El ejemplo siguiente muestra una referencia cruzada ordenada de nombres de datos generada por la opción de compilador XREF . Los números entre paréntesis hacen referencia a las notas después del ejemplo.

```
An "M" preceding a data-name reference indicates that the
data-name is modified by this reference.

     (1)        (2)                            (3)
  Defined   Cross-reference of data-names   References

      265   ABEND-ITEM1
      266   ABEND-ITEM2
      347   ADD-CODE . . . . . .  1102 1162
      381   ADDRESS-ERROR. . . . . .  M1126
      280   AREA-CODE. . . . . .  1236 1261 1324 1345
      382   CITY-ERROR . . . . . .  M1129

        (4)
 Context usage is indicated by the letter preceding a procedure-name
 reference. These letters and their meanings are:
     A = ALTER (procedure-name)
     D = GO TO (procedure-name) DEPENDING ON
     E = End of range of (PERFORM) through (procedure-name)
     G = GO TO (procedure-name)
     P = PERFORM (procedure-name)
     T = (ALTER) TO PROCEED TO (procedure-name)
     U = USE FOR DEBUGGING (procedure-name)

     (5)       (6)                            (7)
  Defined   Cross-reference of procedures   References

      877   000-DO-MAIN-LOGIC
      930   050-CREATE-STL-MAIN-FILE . . .  P879
      982   100-INITIALIZE-PARAGRAPH . . .  P880
     1441   1100-PRINT-I-F-HEADINGS. . . .  P915
     1481   1200-PRINT-I-F-DATA. . . . . .  P916
     1543   1210-GET-MILES-TIME. . . . . .  P1510
     1636   1220-STORE-MILES-TIME. . . . .  P1511
     1652   1230-PRINT-SUB-I-F-DATA. . . .  P1532
     1676   1240-COMPUTE-SUMMARY . . . . .  P1533
     1050   200-EDIT-UPDATE-TRANSACTION. .  P886
     1124   210-EDIT-THE-REST. . . . . .  P1116
     1159   300-UPDATE-COMMUTER-RECORD . .  P888
     1207   310-FORMAT-COMMUTER-RECORD . .  P1164 P1179
     1258   320-PRINT-COMMUTER-RECORD. . .  P1165 P1176 P1182 P1192
     1288   330-PRINT-REPORT . . . . . .  P1178 P1202 P1256 P1280 P1340 P1365 P1369
     1312   400-PRINT-TRANSACTION-ERRORS .  P890
```

Referencia cruzada de nombres de datos:

(1)
Número de línea donde se ha definido el nombre.

(2)
Nombre de datos.

(3)
Números de línea donde se ha utilizado el nombre. Si `M` precede al número de línea, el elemento de datos se ha modificado explícitamente en la ubicación.

Referencias cruzadas de procedimientos:

(4)
Explicaciones de los códigos de uso de contexto para referencias de procedimiento.

(5)
Número de línea donde está definido el nombre-procedimiento.

(6)
Nombre-procedimiento.

(7)
Números de línea donde se hace referencia al procedimiento y el código de uso de contexto para el procedimiento.

"Ejemplo: Salida XREF: referencias cruzadas de nombre de programa" en la página 392
"Ejemplo: Salida XREF: COPY/BASE referencias cruzadas" en la página 392
"Ejemplo: Salida XREF: referencia cruzada incorporada" en la página 393

Ejemplo: Salida XREF: referencias cruzadas de nombre de programa

El ejemplo siguiente muestra una referencia cruzada ordenada de nombres de programa producidos por la opción de compilador XREF . Los números entre paréntesis hacen referencia a las notas que siguen el ejemplo.

```
   (1)       (2)                          (3)
Defined   Cross-reference of programs     References

EXTERNAL   EXTERNAL1. . . . . . . . . . . 25
      2    X. . . . . . . . . . . . . . .  41
     12    X1 . . . . . . . . . . . . . .  33 7
     20    X11. . . . . . . . . . . . . .  25 16
     27    X12. . . . . . . . . . . . . .  32 17
     35    X2 . . . . . . . . . . . . . .  40 8
```

(1)
Número de línea donde se ha definido el nombre de programa. Si el programa es externo, se visualiza la palabra EXTERNAL en lugar de un número de línea de definición.

(2)
Nombre de programa.

(3)
Números de línea en los que se hace referencia al programa.

Ejemplo: Salida XREF: COPY/BASE referencias cruzadas

El ejemplo siguiente muestra una referencia cruzada ordenada de libros de copias a los nombres de biblioteca y nombres de archivo de los libros de copias asociados, producida por la opción de compilador XREF. Los números entre paréntesis hacen referencia a las notas después del ejemplo.

```
        COPY/BASIS cross-reference of text-names, library names and file names

   (1)                          (1)                   (2)
Text-name                    Library-name            File
name
                      (3)                      (3)  (4)
"realxrealyrealzlongxlo>    'thisislongdirecto>    <toryname/
realxrealyrealzlongxname.cpy
"realxrealyrealzlongxlo>    SYSLIB (default)    (5)./cbldir1/
realxrealyrealzlongxname.cpy
"copyA.cpy"                  SYSLIB (default)       ./cbldir1/copyA.cpy
'./copydir2/copyM.cbl'       SYSLIB (default)       ./copydir2/copyM.cbl
'/copyB.cpy'                 SYSLIB                 ./cbldir1/copyB.cpy
'/copydir/copyM.cbl'         SYSLIB                 ./cbldir1/copydir/copyM.cbl
'cbldir1/copyC.cpy'          ALTDD2                 ./cbldir1/copyC.cpy
'copydir/copyM.cbl'          SYSLIB                 ./cbldir1/copydir/copyM.cbl
'copydir2/copyM.cbl'         SYSLIB (default)       ./copydir2/copyM.cbl
'copydir3/stuff.cpy'         ALTDD2                 ./copydir3/stuff.cpy
'stuff.cpy'                  ALTDD                  ./copydir3/stuff.cpy
OTHERDD                      ALTDD2                 .//cbldir1/other.cob
REALXLONGXLONGYNAMEX         SYSLIB (default)       ./REALXLONGXLONGYNAMEX
. . .
```

```
   (5)
./ = /afs/stllp.sanjose.ibm.com/usr1/cobdev/tmross/stuff/subdir
Note: Some names were truncated.   > = truncated on right   < = truncated on left
```

(1)
En la sentencia COPY del origen; por ejemplo, la sentencia COPY correspondiente al quinto elemento de la referencia cruzada anterior sería:

```
COPY '/copyB.cpy' Of SYSLIB
```

(2)
La vía de acceso completa del archivo desde el que se ha copiado el miembro COPY

(3)
El truncamiento de un nombre de texto largo o un nombre de biblioteca a la derecha se marca con un signo mayor que (>).

(4)
El truncamiento de un nombre de archivo largo a la izquierda se marca con un signo menor que (<).

(5)
La parte del directorio de trabajo actual del nombre de archivo se indica mediante **./** en la referencia cruzada. La expansión del nombre del directorio de trabajo actual se muestra en su totalidad debajo de la referencia cruzada.

Referencias relacionadas
“XREF” en la página 306

Ejemplo: Salida XREF: referencia cruzada incorporada

El ejemplo siguiente muestra una referencia cruzada modificada que está incorporada en el listado de origen. La referencia cruzada la genera la opción de compilador XREF .

```
LineID  PL SL  ----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7-|--+----8  Map and Cross Reference
  . . .                                                                                 |
000878                procedure division.                                               |
000879                  000-do-main-logic.                                              |
000880                    display "PROGRAM IGYTCARA - Beginning".                       |
000881                    perform 050-create-stl-main-file.                             | 932 (1)
000882                    perform 100-initialize-paragraph.                             | 984
000883                    read update-transaction-file into ws-transaction-record       | 204 340
000884                        at end                                                    |
000885     1                     set transaction-eof to true                            | 254
000886                    end-read.                                                     |
  . . .                                                                                 |
000984                  100-initialize-paragraph.                                       |
000985                    move spaces to ws-transaction-record                          | IMP 340 (2)
000986                    move spaces to ws-commuter-record                             | IMP 316
000987                    move zeroes to commuter-zipcode                               | IMP 327
000988                    move zeroes to commuter-home-phone                            | IMP 328
000989                    move zeroes to commuter-work-phone                            | IMP 329
000990                    move zeroes to commuter-update-date                           | IMP 333
000991                    open input update-transaction-file                            | 204
000992                        location-file                                             | 193
000993                        i-o commuter-file                                         | 181
000994                        output print-file                                         | 217
  . . .                                                                                 |
001442                1100-print-i-f-headings.                                        |
001443                                                                                  |
001444                    open output print-file.                                       | 217
001445                                                                                  |
001446                    move function when-compiled to when-comp.                     | IFN 698 (2)
001447                    move when-comp (5:2) to compile-month.                        | 698 640
001448                    move when-comp (7:2) to compile-day.                          | 698 642
001449                    move when-comp (3:2) to compile-year.                         | 698 644
001450                                                                                  |
001451                    move function current-date (5:2) to current-month.            | IFN 649
001452                    move function current-date (7:2) to current-day.              | IFN 651
001453                    move function current-date (3:2) to current-year.             | IFN 653
001454                                                                                  |
001455                    write print-record from i-f-header-line-1                     | 222 635
001456                          after new-page.                                         | 138
  . . .                                                                                 |
```

(1)
Número de línea de la definición del nombre de datos o nombre de procedimiento en el programa

(2)
Símbolos de definición especiales:

UND
El nombre de usuario no está definido.

DUP
El nombre de usuario se define más de una vez.

IMP
Nombre definido implícitamente, como registros especiales y constantes figurativas.

IFN
Referencia de función intrínseca.

EXT
Referencia externa.

*****
El nombre de programa no se ha resuelto porque la opción NOCOMPILE está en vigor.

Ejemplo: salida del compilador VBREF

El ejemplo siguiente muestra un listado alfabético de todas las sentencias de un programa y muestra dónde se hace referencia a cada una. El listado lo genera la opción de compilador VBREF .

```
 (1)     (2)                 (3)
2        ACCEPT . . . . . .  1010 1012
2        ADD. . . . . . . .  1290 1306
1        CALL . . . . . . .  1406
5        CLOSE. . . . . . .  898 945 970 1526 1535
20       COMPUTE. . . . . .  1506 1640 1644 1657 1660 1663 1664 1665 1678 1682 1686 1691 1696 1701 1709 1713
                             1718 1723 1728 1733
2        CONTINUE . . . . .  1062 1069
2        DELETE . . . . . .  964 1193
48       DISPLAY. . . . . .  878 906 917 918 919 933 940 942 947 953 960 966 972 996 997 998 999 1003 1006 1037
                             1090 1168 1171 1185 1195 1387 1388 1389 1390 1391 1392 1393 1401 1402 1403 1404
                             1405 1433 1485 1486 1492 1497 1498 1520 1521 1528 1529 1624
2        EVALUATE . . . . .  1161 1557
47       IF . . . . . . . .  887 905 932 939 946 952 959 965 971 993 1002 1036 1051 1054 1071 1074 1077 1089
                             1102 1111 1115 1125 1128 1131 1134 1137 1141 1145 1148 1151 1167 1184 1194 1240
                             1247 1265 1272 1289 1321 1330 1339 1351 1361 1484 1496 1519 1527
183      MOVE . . . . . . .  907 937 957 983 984 985 986 987 988 1004 1011 1013 1025 1038 1052 1055 1060 1067
                             1072 1075 1078 1079 1080 1081 1082 1083 1091 1103 1112 1126 1129 1132 1135 1139
                             1143 1146 1149 1152 1160 1163 1169 1175 1177 1180 1181 1186 1191 1196 1201 1208
                             1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1229 1230
                             1231 1232 1233 1234 1235 1239 1241 1244 1248 1250 1251 1253 1254 1255 1257 1258
                             1259 1260 1264 1266 1269 1273 1275 1276 1278 1279 1291 1294 1299 1301 1303 1307
                             1313 1314 1315 1316 1317 1318 1319 1320 1322 1323 1327 1328 1331 1333 1334 1336
                             1338 1341 1342 1343 1344 1348 1349 1352 1354 1355 1357 1362 1364 1368 1374 1375
                             1376 1377 1378 1379 1380 1381 1414 1417 1422 1425 1445 1446 1447 1448 1450 1451
                             1452 1457 1464 1489 1502 1507 1508 1509 1517 1551 1561 1566 1571 1576 1581 1586
                             1591 1596 1601 1606 1611 1616 1621 1626 1627 1679 1683 1688 1693 1698 1703 1710
                             1715 1720 1725 1730 1735
5        OPEN . . . . . . .  931 951 989 1443 1483
62       PERFORM. . . . . .  879 880 885 886 888 890 892 908 909 915 916 934 935 941 943 948 949 954 955 961
                             962 967 968 973 974 1000 1005 1008 1023 1039 1092 1093 1116 1164 1165 1170 1172
                             1176 1178 1179 1182 1187 1188 1192 1197 1198 1202 1246 1256 1271 1280 1329 1340
                             1350 1359 1365 1369 1504 1510 1511 1532 1533
8        READ . . . . . . .  881 893 958 1014 1026 1085 1490 1514
1        REWRITE. . . . . .  1183
4        SEARCH . . . . . .  1058 1065 1413 1421
46       SET. . . . . . . .  883 895 1016 1028 1041 1057 1064 1084 1087 1363 1412 1420 1493 1499 1516 1522 1548
                             1550 1559 1560 1564 1565 1569 1570 1574 1575 1579 1580 1584 1585 1589 1590 1594
                             1595 1599 1600 1604 1605 1609 1610 1614 1615 1619 1620 1639 1643
2        STOP . . . . . . .  920 1434
4        STRING . . . . . .  1236 1261 1324 1345
33       WRITE. . . . . . .  938 1166 1292 1293 1295 1296 1297 1298 1300 1302 1305 1454 1459 1462 1465 1467 1469
                             1471 1512 1654 1655 1667 1668 1669 1740 1742 1744 1745 1746 1747 1748 1749 1750
```

(1)
Número de veces que se utiliza la sentencia en el programa

(2)
sentencia

(3)
Números de línea donde se utiliza la sentencia

Depuración con mensajes que tienen información de desplazamiento

Algunos mensajes IWZ incluyen información de desplazamiento que puede utilizar para identificar la línea concreta de un programa que ha fallado.

Acerca de esta tarea

Para utilizar esta información:

Procedimiento

1. Compile el programa con la opción LIST . Este paso produce un archivo de listado de ensamblador, que tiene un sufijo de .wlist.
2. Cuando obtenga un mensaje que incluya un rastreo inverso, busque la información de desplazamiento para el programa COBOL. En el ejemplo siguiente, el programa es TEST y el desplazamiento hexadecimal correspondiente es 0x678 (resaltado en negrita):

```
Traceback:
/opt/ibm/cobol/rte/usr/lib/libcob2_32r.so(+0x66b60)[0xf76f3b60]
/opt/ibm/cobol/rte/usr/lib/libcob2_32r.so(iwzWriteERRmsg+0x17)[0xf76f41f7]
/opt/ibm/cobol/rte/usr/lib/libcob2_32r.so(_iwzcBCD_CONV_Pckd_To_Int4+0x176)[0xf76be7b6]
test.out(TEST+0x678)[0x5655a844]
/lib/libc.so.6(__libc_start_main+0xf3)[0xf74b12d3]
    --- End of call chain ---
IWZ903S  The system detected a data exception.
IWZ901S  Program exits due to severe or critical error.
```

3. Busque en el archivo `.wlist` el desplazamiento hexadecimal. A la derecha del desplazamiento hexadecimal, después de los bytes de instrucciones, se encuentra el número de línea de origen COBOL. En el ejemplo siguiente, el número de línea correspondiente a 0x678 es 174 (resaltado en negrita):

```
000174:            COMPUTE x = FUNCTION FACTORIAL(Packed-Dec-05).
    000669  EC83 6A08               000174        sub   esp, 0x00000008
    00066C  046A                    000174        push  0x00000004
    00066E  0068 0000 E800          000174        push  OFFSET FLAT:LEVEL-01-PACKED-DECIMAL
    000673  00E8 0000 8300          000174        call  OFFSET
FLAT:_iwzcBCD_CONV_Pckd_To_Int4
    000678  C483 8910               000174        add   esp, 0x00000010
```

4. Busque la sentencia en el listado del programa.

Referencias relacionadas
“lista” en la página 287

Depuración de rutinas de ensamblador

Utilice la vista Desensamblado para depurar rutinas de ensamblador. Puesto que las rutinas de ensamblador no tienen información de depuración, el depurador va automáticamente a esta vista.

Acerca de esta tarea

Establezca un punto de interrupción en una sentencia desensamblada en la vista Desensamblado efectuando una doble pulsación en el área de prefijo. De forma predeterminada, el depurador cuando se inicia se ejecuta hasta que llega a la primera sentencia depurable. Para que el depurador se detenga en la primera instrucción de la aplicación (depurable o no), debe utilizar la opción `-i` . Por ejemplo:

```
irmtdbgc –i -qhost=myhost progname
```

Parte 4. Destino de programas COBOL para determinados entornos

Capítulo 17. Programación para un entorno Db2

En general, la codificación para un programa COBOL será la misma si desea que el programa acceda a una base de datos Db2 . Sin embargo, para recuperar, actualizar, insertar y suprimir datos de Db2 y utilizar otros servicios de Db2 , debe utilizar sentencias SQL.

Para comunicarse con Db2, siga estos pasos:

- Compruebe el archivo `cob2.cfg` para asegurarse de que las vías de acceso a la versión de Db2 específica se hayan configurado correctamente. Se recomienda utilizar el mandato `cob2_db2` y la stanza `cob2_db2` correspondiente en el archivo `cob2.cfg` al programar para un entorno Db2 . Para obtener más información, consulte "Modificación de la configuración de compilador predeterminada" en la página 242.
- Asegúrese de que el paquete PAM está instalado.
- Codifique las sentencias SQL que necesite, delimitándolas con sentencias `EXEC SQL` y `END-EXEC` . La sentencia `EXEC SQL CONNECT` se conecta a la base de datos cuando se ejecuta el programa. Para obtener más información sobre las sentencias SQL, consulte Introducción a SQL incorporado en la documentación de Db2 .
- Inicie Db2 si todavía no se ha iniciado antes de compilar el programa.
- Especifique la base de datos que utiliza el programa. El compilador se conectará a esa base de datos durante la compilación.
- Si se está conectando a una base de datos Db2 remota, debe configurar Db2 para habilitar la autenticación a través del sistema operativo. Consulte Visión general del modelo de seguridad deDb2 para obtener más información.
- Establezca las variables de entorno LD_LIBRARY_PATH, NLSPATH, DB2INSTANCEy SYSLIB antes de compilar. Tenga en cuenta que el directorio LD_LIBRARY_PATH especificado debe contener `libdb2.so`, que es el objeto compartido para el coprocesador de Db2 , para que el compilador pueda cargar el coprocesador y utilizarlo durante la compilación.

  A continuación se muestra un ejemplo:

  ```
  export LD_LIBRARY_PATH=/opt/ibm/db2/<Db2_version>/lib64
  export NLSPATH=/opt/ibm/db2/<Db2_version>/msg/%L/%N
  export DB2INSTANCE=db2in115
  export SYSLIB=/opt/ibm/db2/<Db2_version>/include/cobol_a
  ```

  Nota: Los archivos de libro de copias bajo el directorio `cobol_a` sólo se incluyen en Db2 11.5.6. Si está utilizando una versión anterior de Db2 , debe ponerse en contacto con COBOL.Linux.Trial@ca.ibm.com para obtener estos archivos.
- Compilar con la opción de compilador SQL.
- Compilar con la opción de compilador `NODYNAM` .

  Nota: La opción de compilador `NODYNAM` es necesaria para programas que contienen sentencias EXEC CICS o EXEC SQL.

  Si se utilizan sentencias `EXEC SQL` en programas COBOL cargados por una llamada dinámica COBOL, una o más sentencias `EXEC SQL` deben estar en el programa principal. En una aplicación Db2 , el programa principal debe realizar la carga inicial de las bibliotecas de Db2 antes de que cualquier programa llamado dinámicamente intente utilizar dichas bibliotecas.
- Utilice las opciones `-L` y `-l` al enlazar. A continuación se muestra un ejemplo:

  ```
  -L/opt/ibm/db2/<Db2_version>/lib64 -ldb2
  ```

  Nota: Es posible que tenga una referencia no definida a sqlgstrt, sqlgaloc, sqlgstlv, sqlgcall, sqlgstop y otros símbolos como, por ejemplo, `undefined reference to `SQLGSTRT'`. Para resolver el problema, especifique las bibliotecas de Db2 utilizando las opciones `-L` y `-l` al enlazar el programa.

Estas opciones deben aparecer después de los archivos fuente COBOL especificados en la línea de mandatos.

A continuación se muestra un ejemplo:

```
filea.cbl -L/opt/ibm/db2/<Db2_version>/lib64 -ldb2
```

Conceptos relacionados
"CoprocesadorDb2" en la página 401

Tareas relacionadas
"Utilización de archivos de Db2" en la página 151
"Codificación de sentencias SQL" en la página 401
"Conexión a la base de datos" en la página 403
"Compilación con la opción SQL" en la página 403
"Pasar opciones al enlazador" en la página 251

Referencias relacionadas
"Variables de entorno de compilador y tiempo de ejecución" en la página 230
"DYNAM" en la página 281
Consulta de SQL para Db2

Asegurarse de que el paquete PAM está instalado

Es necesario que tenga instalado el paquete PAM para acceder a Db2 con COBOL for Linux en x86. Para comprobar si se ha instalado durante la instalación de Db2 , ejecute este mandato:

```
sudo yum list 'pam'
```

, donde el mandato sudo o convertirse en el usuario root garantiza que tiene el privilegio para ejecutar este mandato.

Si la biblioteca PAM no está instalada al ejecutar el compilador con la opción `-qsql` para convertir las sentencias EXEC SQL, obtendrá el siguiente mensaje IGYDS0220 que indica que el compilador no puede cargar el coprocesador Db2 :

```
The "SQL" compiler option was in effect, but the compiler was unable to load the IBM Db2 SQL
co-processor services module. All "EXEC SQL" statements were discarded.
```

Aunque la falta del paquete PAM es una de las razones por las que obtiene el mensaje IGYDS0220 , otras razones podrían ser las siguientes:

- La variable de entorno LD_LIBRARY_PATH no se ha exportado o no contiene la vía de acceso donde está instalada la biblioteca de coprocesador.
- La biblioteca de coprocesador está dañada.
- No tiene suficientes permisos de sistema de archivos para utilizar la biblioteca de coprocesador.
- Una o más bibliotecas que el coprocesador requiere no están presentes, como por ejemplo el paquete PAM. Si el paquete PAM se instala tal como se ha comprobado con el mandato `yum list` mencionado anteriormente y sigue obteniendo el error, puede comprobar si faltan otras bibliotecas con el mandato siguiente:
  - Para aplicaciones de 64 bits:

    ```
    ldd /opt/ibm/db2/<Db2_version>/lib64/libdb2.so
    ```

  - Para aplicaciones de 32 bits:

    ```
    ldd /opt/ibm/db2/<Db2_version>/lib32/libdb2.so
    ```

Para instalar el paquete PAM:

- En RHEL o SUSE, utilice el mandato siguiente:
  - Para aplicaciones de 64 bits:

    ```
    sudo dnf install pam
    ```

  - Para aplicaciones de 32 bits:

    ```
    sudo dnf install parm.i686
    ```

  , donde `dnf` es el instalador de paquetes predeterminado en RHEL y SUSE.
- En Ubuntu, utilice el mandato siguiente:
  - Para aplicaciones de 64 bits:

    ```
    sudo apt-get install libpam0g
    ```

  - Para aplicaciones de 32 bits:

    ```
    sudo apt-get install libpam0g:i386
    ```

  , donde `apt-get` es el instalador de paquetes predeterminado en Ubuntu.

CoprocesadorDb2

Cuando se utiliza el Db2 coprocesador, el compilador maneja los programas fuente que contienen sentencias de SQL incorporado sin tener que utilizar un precompilador. independiente.

Para utilizar el coprocesador Db2 , especifique la opción de compilador `SQL` .

Cuando el compilador encuentra sentencias SQL en el programa de origen, interactúa con el coprocesador de Db2 . Todo el texto entre las sentencias `EXEC SQL` y `END-EXEC` se pasa al coprocesador. El coprocesador realiza las acciones adecuadas para las sentencias SQL e indica al compilador qué sentencias COBOL nativas generar para ellas.

Determinadas restricciones sobre el uso del lenguaje COBOL que se aplican con el precompilador Db2 no se aplican cuando se utiliza el coprocesador Db2 :

- Puede identificar variables del lenguaje principal utilizadas en sentencias SQL sin utilizar las sentencias `EXEC SQL BEGIN DECLARE SECTION` y `EXEC SQL END DECLARE SECTION` .
- Puede compilar por lotes un archivo fuente que contenga varios programas COBOL no anidados.
- El programa fuente puede contener programas anidados.
- El formato de origen ampliado está totalmente soportado.

Tareas relacionadas
“Compilación con la opción SQL” en la página 403

Referencias relacionadas
“SQL” en la página 298
“FORMATOORIGEN” en la página 298

Codificación de sentencias SQL

Delimite las sentencias SQL con `EXEC SQL` y `END-EXEC`. Los delimitadores `EXEC SQL` y `END-EXEC` deben estar completos cada uno en una línea. No puede continuarlos en varias líneas. No codificar sentencias COBOL dentro de sentencias `EXEC SQL` .

Acerca de esta tarea

Debe tener una sentencia EXEC SQL CONNECT para que cuando se ejecute el programa establezca una conexión con la base de datos. A continuación se muestra un ejemplo:

```
EXEC SQL CONNECT TO dbname END-EXEC
```

También debe realizar estos pasos especiales:

- Codifique una sentencia EXEC SQL INCLUDE para incluir un área de comunicación SQL (SQLCA) en WORKING-STORAGE SECTION o LOCAL-STORAGE SECTION del programa más externo. Se recomienda LOCAL-STORAGE para programas recursivos.
- Defina todas las variables del lenguaje principal que utilice en sentencias SQL en WORKING-STORAGE SECTION, LOCAL-STORAGE SECTIONo LINKAGE SECTION. Sin embargo, no es necesario identificarlos con EXEC SQL BEGIN DECLARE SECTION y EXEC SQL END DECLARE SECTION.

Puede utilizar sentencias SQL incluso para objetos grandes (como BLOB y CLOB) y SQL compuesto.

Tareas relacionadas
"Utilización de archivos y sentencias SQL de Db2 en el mismo programa" en la página 152
"Utilización de SQL INCLUDE con el coprocesador de Db2" en la página 402
"Utilización de elementos binarios en sentencias SQL" en la página 402
"Determinación del éxito de las sentencias SQL" en la página 403

Utilización de SQL INCLUDE con el coprocesador de Db2

Una sentencia INCLUDE de SQL se trata de forma idéntica a una COPY sentencia de COBOL nativa (incluida la vía de acceso de búsqueda y las sufijos de archivo utilizados) cuando se utiliza SQL opción de compilador.

Acerca de esta tarea

Por lo tanto, las dos líneas siguientes se tratan de la misma manera. (El punto que finaliza la sentencia EXEC SQL INCLUDE es obligatorio.)

```
EXEC SQL INCLUDE name END-EXEC.
COPY name.
```

La sentencia *nombre* in an SQL INCLUDE sigue las mismas reglas que las de COPY *nombre-texto* y se procesa de forma idéntica a una sentencia COPY *nombre-texto* que no tiene una frase REPLACING .

COBOL no utiliza la variable de entorno de Db2 DB2INCLUDE para el proceso de SQL INCLUDE . Si utiliza la variable de entorno DB2INCLUDE para el proceso de SQL INCLUDE , puede concatenarla con el valor de la variable de entorno COBOL SYSLIB en el archivo .profile del directorio de inicio o en el indicador del shell de mandatos de a Linux . Por ejemplo:

```
export SYSLIB=$DB2INCLUDE:$SYSLIB
```

Referencias relacionadas
Capítulo 14, "Sentencias de direccionamiento de compilador", en la página 309
sentencia COPY (*COBOL for Linux en x86 Consulta de lenguaje*)

Utilización de elementos binarios en sentencias SQL

Para los elementos de datos binarios que especifique en una sentencia EXEC SQL , puede definir los elementos de datos como USAGE COMP-5 o como USAGE BINARY, COMPo COMP-4.

Acerca de esta tarea

Si define los elementos de datos binarios como USAGE BINARY, COMPo COMP-4, utilice la opción TRUNC(BIN). (Esta técnica puede tener un efecto mayor en el rendimiento que utilizar USAGE COMP-5 en elementos de datos individuales.) Si en su lugar TRUNC(OPT) o TRUNC(STD) está en vigor, el compilador acepta los elementos, pero es posible que los datos no sean válidos debido a las reglas de truncamiento decimal. Debe asegurarse de que el truncamiento no afecta a la validez de los datos.

Conceptos relacionados
"Formatos para datos numéricos" en la página 39

Referencias relacionadas
"TRUNC" en la página 302

Determinación del éxito de las sentencias SQL

Cuando Db2 termina de ejecutar una sentencia SQL, Db2 envía un código de retorno en la estructura SQLCA de SQLCODE y SQLSTATE campos de la estructura SQLCA para indicar si la operación ha sido satisfactoria o ha fallado. En el programa, pruebe el código de retorno y realice las acciones necesarias.

Acerca de esta tarea

Referencias relacionadas
Estructura del área de comunicaciones SQL (SQLCA)
Información de SQLCODE, SQLSTATE y SQLWARN

Conexión a la base de datos

Para poder compilar un programa que contenga sentencias EXEC SQL, el compilador debe poder conectarse a la base de datos que utilizará el programa.

Acerca de esta tarea

Puede especificar la base de datos por cualquiera de estos medios:

- Utilice la subopción DATABASE en la opción SQL . A continuación se muestra un ejemplo:

```
cob2_db2 -q"sql('database dbname')" MYSQL.cbl
```

- Establezca la variable de entorno DB2DBDFT antes de invocar el compilador. A continuación se muestra un ejemplo:

```
export DB2DBDFT=dbname
```

Compilación con la opción SQL

La serie de opción que proporcione en la opción de compilador SQL se pone a disposición del coprocesador de Db2 . Solo el coprocesador de Db2 visualiza el contenido de la serie.

Acerca de esta tarea

Por ejemplo, el siguiente mandato cob2_db2 pasa el nombre de base de datos SAMPLE y las opciones de Db2 USER y USING al coprocesador:

```
cob2_db2 -q"sql('database sample user myname using mypassword')" mysql.cbl. . .
```

El coprocesador de Db2 admite las opciones soportadas por el precompilador de Db2 excepto las siguientes:

- MESSAGES
- NOLINEMACRO
- OPTLEVEL
- OUTPUT
- SQLCA
- TARGET
- WCHARTYPE

Por ejemplo, la subopción **bindfile** es una opción soportada por el coprocesador de Db2 . Esta opción se puede especificar por sí misma `-qsql('bindfile')` o con un nombre para el archivo de vinculación `-qsql('bindfile filename')`.

Tareas relacionadas
“Separación de subopciones de Db2” en la página 404
“Utilización de nombres de paquetes y de archivos de enlace” en la página 404

Referencias relacionadas
“SQL” en la página 298
Mandato PRECOMPILE en la documentación de Db2

Separación de subopciones de Db2

Debido a la concatenación de varias especificaciones de opción SQL , puede separar las subopciones de Db2 (que podrían no caber en una sentencia CBL) en varias sentencias CBL .

Acerca de esta tarea

Las opciones que incluye en la serie de subopciones son acumulativas. El compilador concatena estas subopciones de varios orígenes en el orden en que se especifican. Por ejemplo, supongamos que el archivo de origen mypgm.cbl tiene el código siguiente:

```
cbl . . . SQL("string2") . . .
cbl . . . SQL("string3") . . .
```

Cuando emite el mandato `cob2 mypgm.cbl -q:"SQL('string1')"`, el compilador pasa la siguiente serie de subopción al coprocesador de Db2 :

```
"string1 string2 string3"
```

Las series concatenadas se delimitan con espacios únicos. Si el compilador encuentra varias instancias de la misma subopción SQL , la última especificación de dicha subopción en la serie concatenada entrará en vigor. El compilador limita la longitud de la serie de subopción Db2 concatenada a 4 KB.

Utilización de nombres de paquetes y de archivos de enlace

Dos de las subopciones que puede especificar con la opción SQL son `package name` y `bindfile name`. Si no especifica estos nombres, los nombres por omisión se construyen basándose en el nombre de archivo fuente para una compilación no por lotes o en el primer programa para una compilación por lotes.

Acerca de esta tarea

Para los programas no anidados posteriores de una compilación por lotes, los nombres se basan en el PROGRAM-ID de cada programa.

Para el nombre de paquete, el nombre base (el nombre de archivo de origen o PROGRAM-ID) se modifica de la forma siguiente:

- Los nombres de más de ocho caracteres se truncan a ocho caracteres.

- Las letras minúsculas se convierten a mayúsculas.
- Cualquier carácter que no sea A-Z, 0-9 o _ (subrayado) se cambia a 0.
- Si el primer carácter no es alfabético, se cambia a A.

Por lo tanto, si el nombre base es `9123aB-cd`, el nombre del paquete es `A123AB0C`.

Para bindfile-name, se añade el sufijo .bnd al nombre base. A menos que se especifique explícitamente, el nombre de archivo es relativo al directorio actual.

Creación de procedimientos almacenados externos COBOL en Db2

Utilice la opción `PGMNAME(MIXED)` para crear los procedimientos almacenados COBOL y enlazarlos con la opción `-shared` para producir un objeto compartido. Puede utilizar el mandato `cob2_db2` para realizar la compilación y el enlace al mismo tiempo o por separado. Para enlazar un programa COBOL, debe utilizar siempre `cob2_db2` en lugar de utilizar `gcc` o `ld` directamente.

El párrafo PROGRAM-ID en el origen COBOL para el procedimiento almacenado debe coincidir con el nombre, incluidas las mayúsculas y minúsculas, del objeto compartido que cree en el sistema de archivos al compilar y enlazar el programa.

Especifique la vía de acceso absoluta al objeto compartido en la cláusula EXTERNAL NAME de la sentencia CREATE PROCEDURE. Si el objeto compartido de procedimiento almacenado está en esta vía de acceso:

```
/home/jdoe/db2sp/storedProc1
```

, debe especificar esta vía de acceso en la cláusula EXTERNAL NAME.

Capítulo 18. Desarrollo de programas COBOL para CICS

Puede escribir aplicaciones CICS en COBOL y ejecutarlas en una estación de trabajo Linux utilizando CICS TX o TXSeries. El programa de utilidad `cicstcl` incluido con CICS TX y TXSeries realiza la conversión, compila el programa convertido y enlaza el objeto resultante invocando `cob2` para realizar la compilación y el enlace, pasando la opción `-qcics` . Se recomienda utilizar el programa de utilidad `cicstcl` para escribir aplicaciones CICS . De forma alternativa, puede utilizar directamente el mandato de compilador `cob2` o `cob2_cics` con la opción `-qcics` . No hay ninguna diferencia en la funcionalidad entre los dos métodos.

Acerca de esta tarea

Nota: Para utilizar COBOL for Linux en x86 con TXSeries 9.1, debe tener instalado TXSeries 9.1 PTF2 y arreglo temporal 9.1.0.2-TXSeries-Linux-IF052 . Para obtener más información, consulte Requisitos del sistema para IBM TXSeries for Multiplatforms V9.1 para Linux en x86.

Para preparar aplicaciones COBOL para que se ejecuten en CICS, siga estos pasos:

Procedimiento

1. Asegúrese de que el administrador de CICS haya modificado el archivo de entorno de la región para establecer las variables de entorno COBPATH, LD_LIBRARY_PATH y NLSPATH para incluir el directorio de tiempo de ejecución:

   ```
   export COBPATH=<dynamically accessed program dir>:$COBPATH
   export NLSPATH=<CICS install dir>/msg/%L/%N:$NLSPATH
   export LD_LIBRARY_PATH=<CICS install dir>/lib:$LD_LIBRARY_PATH
   ```

   Además, asegúrese de que se haya otorgado acceso a la región CICS al directorio de tiempo de ejecución.

   El archivo de entorno es `/var/cics_regions/`*xxxxxxxx*`/environment` (donde *xxxxxxxx* es el nombre de la región).
2. Compruebe el archivo `cob2.cfg` para asegurarse de que las vías de acceso a la versión de CICS específica se hayan configurado correctamente. Se recomienda utilizar el mandato **`cob2_cics`** y la stanza **`cob2_cics`** correspondiente en el archivo `cob2.cfg` al desarrollar programas COBOL para CICS. Para obtener más información, consulte “Modificación de la configuración de compilador predeterminada” en la página 242.
3. Cree la aplicación utilizando un editor para realizar las tareas siguientes:
   - Codifique el programa utilizando sentencias COBOL y mandatos CICS .
   - Cree libros de copias COBOL.
   - Cree las correlaciones de pantalla de CICS que utiliza el programa.
4. Utilice el mandato `cicsmap` para procesar los mapas de pantalla.
5. Utilice el mandato `cicstcl` para convertir los mandatos CICS con un conversor CICS integrado y para compilar y enlazar el programa. Si no especifica una extensión de archivo, `cicstcl` utiliza de forma predeterminada la extensión de archivo de origen COBOL `.cbl`.

   Los ejemplos siguientes muestran cómo utilizar el mandato `cicstcl` para convertir, compilar y enlazar un programa COBOL de ejemplo APPLCOB que se ejecuta en TXSeries o CICS TX. El mandato `cicstcl` genera el módulo de salida con la extensión `.ibmcob` .

- Para compilar, traducir y editar enlaces de una aplicación CICS COBOL, utilice la opción `-l IBMCOB` para especificar el lenguaje fuente como IBM COBOL, y la extensión de archivo de programa COBOL de CICS puede ser `.ccp` o `.cbl`:

```
cicstcl -l IBMCOB APPLCOB.ccp
```

```
cicstcl -l IBMCOB APPLCOB.cbl
```

- Para compilar, convertir y editar enlaces de una aplicación CICS COBOL que se va a utilizar para depurar utilizando CEDF, utilice la opción `-e` . Para visualizar números de línea de sentencia CICS , utilice la opción `-d` :

```
cicstcl -e -l IBMCOB APPLCOB.cbl
```

```
cicstcl -e -d -l IBMCOB APPLCOB.cbl
```

- Para compilar, convertir y editar con enlaces una aplicación CICS COBOL que se utilizará para depurar, utilice la opción `-a` :

```
cicstcl -a -l IBMCOB APPLCOB.cbl
```

- Para compilar, convertir y editar enlaces de una aplicación COBOL CICS especificando la vía de acceso COPYBOOK, establezca la variable de entorno SYSLIB para especificar el directorio de la vía de acceso COPYBOOK de origen COBOL y, a continuación, utilice el mandato `cicstcl` :

```
export SYSLIB="/program_copybook_path"
```

```
cicstcl -l IBMCOB APPLCOB.cbl
```

- Para compilar, convertir y editar enlaces de una aplicación CICS COBOL enlazando estáticamente un módulo COBOL, establezca la variable de entorno USERLIB para especificar la vía de acceso del módulo de objeto compilado y, a continuación, utilice el mandato `cicstcl` :

```
export USERLIB="cobol_module.o"
```

```
cicstcl -l IBMCOB APPLCOB.cbl
```

Nota: Si desea compilar y enlazar un programa CICS COBOL sin utilizar el mandato `cicstcl` , utilice el mandato de compilador `cob2` o `cob2_cics` con la opción `-qcics` . No hay ninguna diferencia en la funcionalidad entre los dos métodos. El ejemplo siguiente muestra cómo compilar y enlazar una aplicación CICS COBOL utilizando el mandato `cob2` :

```
cob2 -qNOTHREAD -I/opt/ibm/cics/include -qcics -o APPLCOB.ibmcob APPLCOB.cbl -L/opt/ibm/
cics/lib -lcicsprIBMCOB
```

Para obtener información detallada sobre el uso del mandato `cicstcl` , consulte "cicstcl" en la documentación de TXSeries for Multiplatforms o "cicstcl" en la documentación de CICS TX.

6. Defina los recursos para la aplicación, como por ejemplo transacciones, programas de aplicación y archivos, en la región CICS .

   Se necesita autorización de administrador de CICS para realizar estas acciones.
7. Acceda a la región CICS , por ejemplo, utilizando el mandato `cicsterm` .
8. Ejecute la aplicación especificando el ID de transacción de cuatro caracteres que está asociado con la aplicación.

Qué hacer a continuación

Conceptos relacionados
“Conversor integrado de CICS” en la página 414

Tareas relacionadas
“Codificación de programas COBOL para ejecutarlos en CICS” en la página 409
“Compilación y ejecución de programas CICS” en la página 413
“Depuración de programas CICS” en la página 415
Documentación deTXSeries for Multiplatforms
IBM Documentación de CICS TX

Codificación de programas COBOL para ejecutarlos en CICS

Para codificar un programa para que se ejecute bajo CICS, codifique los mandatos CICS en PROCEDURE DIVISION utilizando el formato de mandato EXEC CICS .

Acerca de esta tarea

```
EXEC CICS command-name command-options
END-EXEC
```

Los mandatos CICS tienen el formato básico que se muestra anteriormente. En los mandatos EXEC , utilice el espacio como separador de palabras; no utilice una coma ni un punto y coma. No codifique sentencias COBOL en mandatos EXEC CICS .

En general, el lenguaje COBOL está soportado en un entorno CICS . Sin embargo, existen restricciones y consideraciones que debe tener en cuenta cuando codifique programas COBOL para que se ejecuten en TXSeries o CICS TX.

Restricciones:

- Los archivos de Db2 que interoperarán con TXSeries o CICS TX se deben crear con FILEMODE(SMALL) en vigor.
- La programación orientada a objetos y la interoperatividad con Java no están soportadas.
- El programa fuente no debe contener ningún programa anidado.
- Los programas COBOL que se ejecutarán bajo TXSeries o CICS TX deben ser de 32 bits.

No utilice EXEC, CICS, o END-EXEC como nombres de variable y no utilice parámetros especificados por el usuario para el programa principal. Además, se recomienda no utilizar ninguno de los siguientes elementos de lenguaje COBOL:

- Entrada FILE-CONTROL en ENVIRONMENT DIVISION
- FILE SECTION en el DATA DIVISION
- USE declarativas, excepto USE FOR DEBUGGING

Las siguientes sentencias COBOL tampoco se recomiendan para su uso en un entorno CICS :

- ACCEPT Formato 1
- CLOSE
- DELETE
- DISPLAY UPON CONSOLE, DISPLAY UPON SYSPUNCH
- MERGE
- OPEN
- READ
- REWRITE
- SORT
- START
- STOP *literal*
- WRITE

Aparte de algunas formas de la sentencia ACCEPT , el sistema principal CICS no da soporte a ninguno de los elementos de lenguaje COBOL de la lista anterior. Si utiliza alguno de estos elementos, tenga en cuenta las limitaciones siguientes:

- El programa no es completamente portátil al entorno CICS del sistema principal.
- En el caso de una anomalía de CICS , no será posible una restitución (restauración de los recursos que están asociados con la tarea anómala) para los recursos que se han actualizado utilizando las sentencias anteriores.

Restricción: No hay soporte de formato de datos de host IBM Z para programas COBOL convertidos por el conversor de separado o integrado CICS y ejecutados en TXSeries o CICS TX.

Tareas relacionadas
"Obtención de la fecha del sistema en CICS" en la página 410
"Realización de llamadas dinámicas en CICS" en la página 410
"Acceso a datos SFS" en la página 413
"Llamada entre COBOL y C/C++ en CICS" en la página 413

Referencias relacionadas
"Sistema de archivos Db2" en la página 123
"ADDR" en la página 268
Apéndice A, "Consideraciones sobre el formato de datos de host de IBM Z", en la página 549

Obtención de la fecha del sistema en CICS

Para recuperar la fecha del sistema en un programa CICS , utilice una sentencia format-2 ACCEPT o la función intrínseca CURRENT-DATE .

Acerca de esta tarea

Puede utilizar cualquiera de estas format-2 ACCEPT sentencias en CICS para obtener la fecha del sistema:

- ACCEPT *identifier-2* FROM DATE (año de dos dígitos)
- ACCEPT *identifier-2* FROM DATE YYYYMMDD
- ACCEPT *identifier-2* FROM DAY (año de dos dígitos)
- ACCEPT *identifier-2* FROM DAY YYYYDDD
- ACCEPT *identifier-2* FROM DAY-OF-WEEK (entero de un dígito, donde 1 representa el lunes)

Puede utilizar esta sentencia format-2 ACCEPT en CICS para obtener la hora del sistema:

- ACCEPT *identifier-2* FROM TIME

De forma alternativa, puede utilizar la función intrínseca CURRENT-DATE , que también puede proporcionar la hora.

Estos métodos funcionan en entornos CICS y noCICS .

No utilice una sentencia format-1 ACCEPT en un programa CICS .

Tareas relacionadas
"Asignación de entrada desde una pantalla o archivo (ACCEPT)" en la página 31

Referencias relacionadas
CURRENT-DATE (*COBOL for Linux en x86 Consulta de lenguaje*)

Realización de llamadas dinámicas en CICS

Puede utilizar sentencias CALL *identifier* para realizar llamadas dinámicas de en el entorno CICS . Sin embargo, debe establecer la variable de entorno COBPATH correctamente. También debe asegurarse de que el módulo llamado tiene el nombre correcto.

Acerca de esta tarea

Considere el ejemplo siguiente, en el que `alpha` es un programa COBOL que contiene sentencias CICS :

```
WORKING-STORAGE SECTION.
01  WS-COMMAREA   PIC 9    VALUE ZERO.
77   SUBPNAME     PIC X(8) VALUE SPACES
. . .
PROCEDURE DIVISION.
    MOVE 'alpha' TO SUBPNAME.
    CALL SUBPNAME USING DFHEIBLK, DFHCOMMAREA, WS-COMMAREA.
```

Debe pasar los bloques de control de CICS `DFHEIBLK` y `DFHCOMMAREA` (como se muestra anteriormente) a `alpha`.

El origen de `alpha` está en el archivo `alpha.ccp`. Utilice el mandato `cicstcl` para convertir, compilar y enlazar `alpha.ccp`. COBOL toma como valor predeterminado los nombres en mayúsculas. Por lo tanto, a menos que cambie este valor predeterminado utilizando la opción de compilador `PGMNAME(MIXED)` , debe nombrar el archivo fuente `ALPHA.ccp` (no `alpha.ccp`) para producir `ALPHA.ibmcob` (no `alpha.ibmcob`).

Supongamos que la región CICS se denomina `green`. A continuación, el archivo `ALPHA.ibmcob` debe copiarse en `/var/cics_regions/green/bin`y el Mandato `cicsadd` para añadir una nueva definición de recurso debe utilizarse para definir ALPHA como un programa CICS . El personal de instalación debe añadir la línea siguiente al archivo `/var/cics_regions/green/environment`:

```
COBPATH=/var/cics_regions/green/bin
```

A continuación, el personal debe concluir la región CICS `green` y reiniciarla. Si coloca los programas llamados dinámicamente en algún otro directorio, asegúrese de que el personal de instalación añade ese directorio a COBPATH y que los servidores CICS tienen permiso para acceder a ese directorio.

Tareas relacionadas
“Realización de llamadas dinámicas a bibliotecas compartidas en CICS” en la página 411
“Ajuste del rendimiento de llamadas dinámicas en CICS” en la página 412

Referencias relacionadas
“Variables de entorno de compilador y tiempo de ejecución” en la página 230

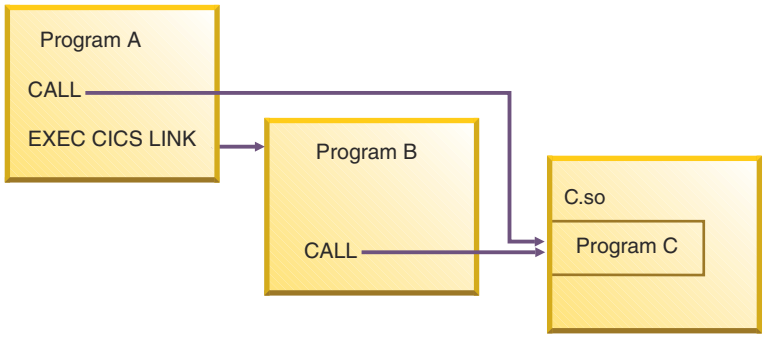
Realización de llamadas dinámicas a bibliotecas compartidas en CICS

Si tiene una biblioteca compartida que contiene uno o más programas COBOL, no la utilice en más de una unidad de ejecución dentro de la misma transacción CICS ; de lo contrario, los resultados son imprevisibles.

Acerca de esta tarea

La figura siguiente muestra una transacción CICS en la que se llama al mismo subprograma desde dos unidades de ejecución diferentes:

- El programa A llama al programa C (en `C.so`).
- El programa A enlaza con el programa B utilizando un mandato `EXEC CICS LINK` . Esta combinación se convierte en una nueva unidad de ejecución dentro de la misma transacción.
- El programa B llama al programa C (en `C.so`).



Los programas A y B comparten la misma copia del Programa C, y cualquier cambio en su estado afecta a ambos programas.

En el entorno CICS , los programas de una biblioteca compartida se inicializan (ya sea mediante la opción de compilador WSCLEAR o mediante la inicialización de la cláusula VALUE) sólo en la primera llamada dentro de una unidad de ejecución. Si se llama a un subprograma COBOL más de una vez desde el mismo programa principal o desde uno diferente, el subprograma se inicializa sólo en la primera llamada.

Si necesita que el subprograma se inicialice en la primera llamada de cada programa principal, enlace estáticamente una copia separada del subprograma con cada programa de llamada. Si necesita que el subprograma se inicialice en cada llamada, utilice uno de los métodos siguientes:

- Coloque los datos que se van a reinicializar en el LOCAL-STORAGE SECTION del subprograma en lugar de en el WORKING-STORAGE SECTION. Esta ubicación sólo afecta a la inicialización mediante cláusulas VALUE , no mediante WSCLEAR.
- Utilice CANCEL para cancelar el subprograma después de cada uso para que la siguiente llamada sea al programa en su estado inicial.
- Añada el atributo INITIAL al subprograma.

Tareas relacionadas
Capítulo 24, “Utilización de bibliotecas compartidas”, en la página 489

Referencias relacionadas
“WCLEAR” en la página 305

Ajuste del rendimiento de llamadas dinámicas en CICS

El rendimiento de las transacciones CICS persistentes se mejora de forma predeterminada en las aplicaciones COBOL para Linux mediante el almacenamiento en memoria caché de módulos.

Acerca de esta tarea

Para habilitar o inhabilitar el almacenamiento en memoria caché de módulos, establezca las variables de entorno siguientes:

```
export COBOL_CPM_CACHE=0    ## To disable caching
export COBOL_CPM_CACHE=1    ## To enable caching
```

Si no se especifica la variable de entorno COBOL_CPM_CACHE, los valores predeterminados son los siguientes:

- Para una transacción CICS, el almacenamiento en memoria caché está habilitado de forma predeterminada.
- Para un programa no CICS, el almacenamiento en memoria caché se controla automáticamente; el tiempo de ejecución COBOL decide si se debe habilitar el almacenamiento en memoria caché y cuándo.

Cuando un módulo se almacena en memoria caché, su semántica de ejecución puede cambiar, porque sus elementos de datos WORKING-STORAGE que no tienen cláusulas VALUE y para los que no hay sentencias de inicialización explícitas permanecen en el estado de última utilización. Si un programa en

memoria no se almacena en memoria caché, sino que se vuelve a cargar desde el disco, se perderá el último estado utilizado de estas variables no inicializadas.

No se base en el valor de los elementos de datos no inicializados. Inicialice estos elementos de datos de forma explícita o utilice temporalmente la opción de compilador WSCLEAR para borrar WORKING-STORAGE a ceros binarios cada vez que se especifique el programa.

WSCLEAR puede afectar al rendimiento porque la opción aumenta el tiempo y posiblemente el espacio necesario para la inicialización del programa. Para obtener el mejor rendimiento cuando el almacenamiento en memoria caché de módulos está en vigor, revise el código de la aplicación y decida si se inicializan explícitamente las variables.

Referencias relacionadas
"WCLEAR" en la página 305

Acceso a datos SFS

Si el programa no se ejecuta en CICS, puede acceder a los archivos SFS a través del sistema de archivos SFS (el sistema de archivos predeterminado utilizado por TXSeries o CICS TX).

Acerca de esta tarea

Tareas relacionadas
"Identificación de archivos" en la página 117
"Identificación de archivos SFS" en la página 120
"Utilización de archivos SFS" en la página 156

Referencias relacionadas
"Sistema de archivos SFS" en la página 126

Llamada entre COBOL y C/C++ en CICS

Puede realizar una llamada bajo CICS desde un programa COBOL a C/C++ o desde un programa C/C++ a un programa COBOL sólo si el programa llamado no contiene ningún mandato CICS . (El programa de llamada puede contener mandatos CICS .)

Acerca de esta tarea

Los programas COBOL pueden emitir un mandato EXEC CICS LINK o EXEC CICS XCTL a un programa C/C++ independientemente de si el programa C/C++ contiene mandatos CICS . Por lo tanto, si el programa COBOL llama a un programa C/C++ que contiene mandatos CICS , utilice EXEC CICS LINK o EXEC CICS XCTL en lugar de la sentencia COBOL CALL .

Tareas relacionadas
"Llamada entre programas COBOL y C/C++" en la página 464

Compilación y ejecución de programas CICS

Los programas Para compilar programas COBOL para Linux TXSeries o CICS TX , utilice el mandato cob2 o cob2_cics .

Acerca de esta tarea

TRUNC(BIN) es una opción de compilador recomendada para un programa COBOL que se ejecutará bajo CICS. Sin embargo, si está seguro de que los valores no truncados de los elementos de datos BINARY, COMPy COMP-4 se ajustan a sus especificaciones PICTURE , es posible que pueda mejorar el rendimiento del programa utilizando TRUNC(OPT).

Puede utilizar un elemento de datos COMP-5 en lugar de un elemento de datos BINARY, COMPo COMP-4 como argumento de mandato EXEC CICS . Los elementos de datos de COMP-5 se tratan como BINARY, COMP, o COMP-4 si TRUNC(BIN) está en vigor.

Debe utilizar la opción de compilador PGMNAME(MIXED) para los programas que utilizan CICS Client.

No utilice la opción de compilador DYNAM o ADDR(64) cuando convierta un programa COBOL (utilizando el conversor separado o integrado CICS). Todas las demás opciones de compilador COBOL están soportadas.

Resultados

Opciones de tiempo de ejecución: Utilice la opción de tiempo de ejecución FILESYS para especificar el sistema de archivos predeterminado si no se ha especificado ningún sistema de archivos para un archivo mediante una cláusula ASSIGN .

Conceptos relacionados
“Conversor integrado de CICS” en la página 414

Tareas relacionadas
“Compilación desde la línea de mandatos” en la página 240
Documentación deTXSeries for Multiplatforms
Documentación de CICS TX

Referencias relacionadas
“opciones de compilador” en la página 265
“Opciones de compilador en conflicto” en la página 267
“ARCHIVOS” en la página 317

Conversor integrado de CICS

Cuando compila un programa COBOL utilizando la opción de compilador CICS , el compilador COBOL trabaja con el conversor CICS integrado para manejar tanto las sentencias COBOL nativas como las sentencias CICS incorporadas en el programa fuente.

Cuando el compilador encuentra sentencias CICS y en otros puntos significativos del programa de origen, el compilador interactúa con el conversor CICS integrado. Todo el texto entre las sentencias EXEC CICS y END-EXEC se pasa al conversor. El conversor realiza las acciones adecuadas y, a continuación, vuelve al compilador, normalmente indicando qué sentencias de lenguaje nativo generar.

Si compila el programa COBOL utilizando el mandato cicstcl , utiliza el conversor CICS integrado. El mandato cicstcl invoca el compilador con las subopciones adecuadas de la opción de compilador CICS .

Aunque todavía puede convertir sentencias CICS incorporadas por separado, se recomienda que utilice en su lugar el conversor CICS integrado. Ciertas restricciones que se aplican cuando se utiliza el conversor separado no se aplican cuando se utiliza el conversor integrado, y el uso del conversor integrado proporciona varias ventajas:

- Puede utilizar para depurar el origen original en lugar del origen expandido que genera el conversor CICS independiente.
- No es necesario convertir por separado las sentencias EXEC CICS que están en libros de copias.
- No hay archivo intermedio para una versión traducida pero no compilada del programa fuente.
- Sólo se genera un listado de salida en lugar de dos.
- Las sentencias REPLACE pueden afectar a las sentencias EXEC CICS .
- Puede compilar programas que contengan sentencias CICS en una compilación por lotes (compilación de una secuencia de programas).
- El formato de origen ampliado está totalmente soportado.

Tareas relacionadas
“Codificación de programas COBOL para ejecutarlos en CICS” en la página 409
“Compilación y ejecución de programas CICS” en la página 413

Referencias relacionadas
“CICS” en la página 274
“FORMATOORIGEN” en la página 298

Depuración de programas CICS

Si compila programas CICS utilizando el conversor integrado, puede depurar los programas en el nivel de origen original en lugar de depurar el origen expandido que proporciona el conversor CICS independiente.

Acerca de esta tarea

Si utiliza el conversor CICS independiente, primero convierta los programas CICS en COBOL. A continuación, puede depurar los programas COBOL resultantes de la misma forma que depura cualquier otro programa COBOL.

Puede depurar programas CICS utilizando IBM Debug for Linux en x86 que se suministra con COBOL para Linux. Asegúrese de indicar al compilador que genere la información simbólica que utiliza el depurador.

Conceptos relacionados
Capítulo 16, “depuración”, en la página 319
“Conversor integrado de CICS” en la página 414

Tareas relacionadas
“Compilación desde la línea de mandatos” en la página 240
Documentación deTXSeries for MultiplatformsDocumentación de CICS TX

Parte 5. Utilización de XML y COBOL juntos

Capítulo 19. Procesando entrada XML

Puede procesar la entrada XML en un programa COBOL utilizando la sentencia XML PARSE .

Acerca de esta tarea

La sentencia XML PARSE es la interfaz de lenguaje COBOL con el analizador XML de alta velocidad que forma parte del tiempo de ejecución COBOL.

El proceso de entrada XML implica pasar el control entre el analizador XML y un procedimiento de proceso en el que se manejan los sucesos del analizador.

Utilice los siguientes recursos COBOL para procesar la entrada XML:

- La sentencia XML PARSE para iniciar el análisis de XML y para identificar el documento XML de origen y el procedimiento de proceso
- El procedimiento de proceso para controlar el análisis, es decir, recibir y procesar sucesos XML y fragmentos de documento asociados, y volver al analizador para continuar el proceso
- Registros especiales para intercambiar información entre el analizador y el procedimiento de proceso:
  - XML-CODE para recibir el estado del análisis XML y, en algunos casos, para devolver información al analizador
  - XML-EVENT para recibir el nombre de cada suceso XML del analizador
  - XML-NTEXT para recibir fragmentos de documento XML que se devuelven como datos de caracteres nacionales
  - XML-TEXT para recibir fragmentos de documento que se devuelven como datos alfanuméricos

Conceptos relacionados
“Analizador XML en COBOL” en la página 419

Tareas relacionadas
“Acceso a documentos XML” en la página 420
“Análisis de documentos XML” en la página 421
“Manejo de excepciones XML PARSE” en la página 430
“Terminando análisis XML” en la página 433

Referencias relacionadas
“La codificación de documentos XML” en la página 426
Apéndice D, “Material de referencia XML”, en la página 597
XML (Extensible Markup Language)

Analizador XML en COBOL

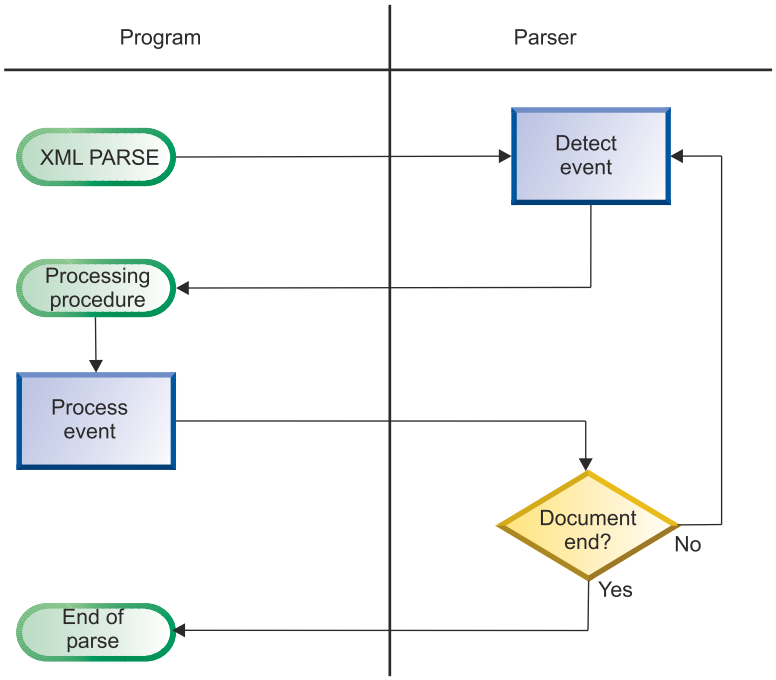
COBOL para Linux proporciona una interfaz basada en sucesos que le permite analizar documentos XML y transformarlos en estructuras de datos COBOL.

El analizador XML busca fragmentos dentro del documento XML de origen y el procedimiento de proceso actúa sobre dichos fragmentos. Los fragmentos se asocian con sucesos XML; codifica el procedimiento de proceso para manejar cada suceso XML.

La ejecución de la sentencia XML PARSE inicia el análisis y establece el procedimiento de proceso con el analizador. El analizador transfiere el control al procedimiento de proceso para cada suceso XML que detecta al procesar el documento. Después de procesar el suceso, el procedimiento de proceso devuelve automáticamente el control al analizador. Cada retorno normal del procedimiento de proceso hace que el analizador continúe analizando el documento XML para notificar el siguiente suceso. A lo largo de esta operación, el control pasa de un lado a otro entre el analizador y el procedimiento de proceso.

En la sentencia XML PARSE , también puede especificar dos sentencias imperativas a las que desea que se pase el control al final del análisis: una si se produce una finalización normal y la otra si existe una condición de excepción.

La figura siguiente muestra una visión general de alto nivel del intercambio básico de control entre el analizador y el programa COBOL:



Normalmente, el análisis continúa hasta que se ha analizado todo el documento XML.

El analizador XML comprueba los documentos XML en busca de la mayoría de los aspectos de buena calidad. Un documento está *bien formado* si se ajusta a la sintaxis XML en *Especificación XML* y sigue algunas reglas adicionales como el uso adecuado de códigos finales y la exclusividad de los nombres de atributo.

Conceptos relacionados
"Codificación de documento de entrada XML" en la página 427

Tareas relacionadas
"Análisis de documentos XML" en la página 421
"Manejo de excepciones XML PARSE" en la página 430
"Terminando análisis XML" en la página 433

Referencias relacionadas
"La codificación de documentos XML" en la página 426
"Conformidad XML" en la página 606
Especificación XML

Acceso a documentos XML

Antes de poder analizar un documento XML utilizando una sentencia XML PARSE , debe hacer que el documento esté disponible para el programa. Los métodos comunes para adquirir un documento XML son de un parámetro a su programa o leyendo el documento de un archivo.

Acerca de esta tarea

Si el documento XML que desea analizar está contenido en un archivo, utilice los recursos COBOL ordinarios para colocar el documento en un elemento de datos del programa:

- Una entrada FILE-CONTROL para definir el archivo en el programa.

- Una sentencia OPEN para abrir el archivo.
- READ sentencias para leer todos los registros del archivo en un elemento de datos (ya sea un elemento elemental de categoría alfanumérica o nacional, o un grupo alfanumérico o nacional). Puede definir el elemento de datos en WORKING-STORAGE SECTION o LOCAL-STORAGE SECTION.
- Opcionalmente, la sentencia STRING para encadenar todos los registros separados en una corriente continua, para eliminar espacios en blanco extraños y para manejar registros de longitud variable.

Análisis de documentos XML

Para analizar documentos XML, utilice la sentencia XML PARSE , especificando el documento XML que se va a analizar y el procedimiento de proceso para manejar sucesos XML que se producen durante el análisis, tal como se muestra en el fragmento de código siguiente.

Acerca de esta tarea

```
XML PARSE xml-document
    PROCESSING PROCEDURE xml-event-handler
  ON EXCEPTION
     DISPLAY 'XML document error ' XML-CODE
     STOP RUN
  NOT ON EXCEPTION
     DISPLAY 'XML document was successfully parsed.'
END-XML
```

En la sentencia XML PARSE , primero debe identificar el *elemento de datos de análisis* (xml-document en el ejemplo anterior) que contiene la corriente de caracteres del documento XML. En DATA DIVISION, defina el elemento de datos de análisis como un elemento de datos elemental de categoría nacional o como un elemento de grupo nacional si la codificación del documento es Unicode UTF-16; de lo contrario, defina el elemento de datos de análisis como un elemento de datos alfanumérico elemental o un elemento de grupo alfanumérico:

- Si el elemento de datos de análisis es nacional, el documento XML debe estar codificado en UTF-16 en formato little-endian.
- Si el elemento de datos de análisis es alfanumérico, su contenido debe codificarse en una de las páginas de códigos soportadas descritas en la referencia relacionada sobre la codificación de documentos XML.

A continuación, especifique el nombre del procedimiento de proceso (xml-event-handler en el ejemplo anterior) que debe manejar los sucesos XML que se producen durante el análisis del documento.

Además, puede especificar una o las dos frases opcionales siguientes (tal como se muestra en el fragmento anterior) para indicar la acción que se debe realizar después de que finalice el análisis:

- ON EXCEPTION, para recibir el control si se produce una excepción no manejada durante el análisis
- NOT ON EXCEPTION, para recibir el control de lo contrario

Puede finalizar la sentencia XML PARSE con el terminador de ámbito explícito END-XML. Utilice END-XML para anidar una sentencia XML PARSE que utilice la frase ON EXCEPTION o NOT ON EXCEPTION en una sentencia condicional.

El analizador pasa el control al procedimiento de proceso para cada suceso XML. El control vuelve al analizador al final del procedimiento de proceso. Este intercambio de control entre el analizador XML y el procedimiento de proceso continúa hasta que se produce uno de los sucesos siguientes:

- Se ha analizado todo el documento XML, tal como indica el suceso END-OF-DOCUMENT .
- El analizador detecta un error en el documento y señala un suceso EXCEPTION , y el procedimiento de proceso no restablece el registro especial XML-CODE a cero antes de volver al analizador.
- El proceso de análisis lo termina deliberadamente el código en el procedimiento de proceso que establece el registro especial XML-CODE en -1 antes de que vuelva al analizador.

Conceptos relacionados
“Sucesos XML” en la página 423
“CÓDIGO XML” en la página 424

Tareas relacionadas
“Especificación de la página de códigos para datos de caracteres” en la página 214
“Escritura de procedimientos para procesar XML” en la página 422
“Análisis de documentos XML codificados en UTF-8” en la página 429

Referencias relacionadas
“La codificación de documentos XML” en la página 426
“Excepciones XML PARSE” en la página 597
sentencia XML PARSE (*COBOL for Linux en x86 Consulta de lenguaje*)

Escritura de procedimientos para procesar XML

En el procedimiento de proceso, codifique sentencias para manejar sucesos XML.

Acerca de esta tarea

Para cada suceso que el analizador encuentra, el analizador pasa información al procedimiento de proceso en varios registros especiales. Utilice el contenido de estos registros especiales para llenar estructuras de datos COBOL y para controlar el proceso.

Examine el registro especial XML-EVENT para determinar qué suceso ha pasado el analizador al procedimiento de proceso. XML-EVENT contiene un nombre de suceso, como por ejemplo 'START-OF-ELEMENT'. Obtenga el texto asociado con el suceso del registro especial XML-TEXT o XML-NTEXT .

Cuando se utilizan en programas anidados, los registros especiales XML se definen implícitamente como GLOBAL en el programa más externo.

Para obtener detalles adicionales sobre los registros especiales XML, consulte la tabla siguiente.

Tabla 38. ***Registros especiales utilizados por el analizador XML***

Registro especial	Definición y uso implícitos	Contenido
XML-EVENT[1]	PICTURE X(30) USAGE DISPLAY VALUE SPACE	El nombre del suceso XML
XML-CODE[2]	PICTURE S9(9) USAGE BINARY VALUE ZERO	Un código de excepción o cero para cada suceso XML
XML-TEXT[1, 3]	Categoría elemento alfanumérico	Texto (correspondiente al suceso que ha encontrado el analizador) del documento XML si especifica un elemento alfanumérico para el XML PARSE identificador
XML-NTEXT[1]	Categoría elemento nacional	Texto (correspondiente al suceso que ha encontrado el analizador) del documento XML si especifica un elemento nacional para el XML PARSE identificador

1. No puede utilizar este registro especial como elemento de datos de recepción.
2. La sentencia XML GENERATE también utiliza XML-CODE. Por lo tanto, si tiene una sentencia XML GENERATE en el procedimiento de proceso, guarde el valor de XML-CODE antes de la sentencia XML GENERATE y restaure el valor guardado después de la sentencia XML GENERATE .
3. El contenido de XML-TEXT tiene la codificación del documento XML de origen: ASCII o UTF-8 si la opción de compilador CHAR(NATIVE) está en vigor; EBCDIC si CHAR(EBCDIC) está en vigor.

Restricciones:

- Un procedimiento de proceso no debe ejecutar directamente una sentencia XML PARSE . Sin embargo, si un procedimiento de proceso pasa el control al programa externo de 1. utilizando la sentencia a CALL , el programa de destino puede ejecutar la misma sentencia XML PARSE o una

sentencia diferente. También puede ejecutar la misma sentencia XML o distintas sentencias XML simultáneamente desde un programa que se ejecuta en varias hebras.

- El rango del procedimiento de proceso no debe provocar la ejecución de ninguna sentencia GOBACK o EXIT PROGRAM , excepto para devolver el control de un método al que una CALL ha pasado el control, respectivamente, que se ejecuta en el rango del procedimiento de proceso.

  Puede codificar una sentencia STOP RUN en un procedimiento de proceso para finalizar la unidad de ejecución.

El compilador inserta un mecanismo de retorno después de la última sentencia en cada procedimiento de proceso.

"Ejemplo: programa para procesar XML" en la página 434

Conceptos relacionados
"Sucesos XML" en la página 423
"CÓDIGO XML" en la página 424
"XML-TEXT y XML-NTEXT" en la página 425

Tareas relacionadas
"Terminando análisis XML" en la página 433

Referencias relacionadas
"char" en la página 272
XML-EVENT (*COBOL for Linux en x86 Consulta de lenguaje*)

Sucesos XML

Un *suceso XML* se produce cuando el analizador XML detecta varias condiciones (como END-OF-INPUT o EXCEPTION) o encuentra fragmentos de documento (como CONTENT-CHARACTERS o START-OF-CDATA-SECTION) al procesar un documento XML.

Para cada suceso que se produce durante el análisis XML, el analizador establece el nombre de suceso asociado en el registro especial XML-EVENT y pasa el registro especial XML-EVENT al procedimiento de proceso. En función del suceso, el analizador establece otros registros especiales para que contengan información adicional sobre el suceso.

En la mayoría de los casos, el analizador establece el registro especial XML-TEXT o XML-NTEXT en el fragmento XML que ha causado el suceso: XML-NTEXT si el documento XML está en un elemento de datos nacional, o si el analizador encuentra una referencia de caracteres; de lo contrario, XML-TEXT.

Cuando el analizador detecta un conflicto de codificación o un error de en el documento, establece XML-EVENT en 'EXCEPTION' y proporciona información adicional sobre la excepción en el registro especial XML-CODE .

Para obtener una descripción detallada del conjunto de sucesos XML, consulte la referencia relacionada sobre XML-EVENT.

Conceptos relacionados
"Analizador XML en COBOL" en la página 419
"CÓDIGO XML" en la página 424

"XML-TEXT y XML-NTEXT" en la página 425

Tareas relacionadas
"Escritura de procedimientos para procesar XML" en la página 422

Referencias relacionadas
"Excepciones XML PARSE" en la página 597
XML-EVENT (*COBOL for Linux en x86 Consulta de lenguaje*)

CÓDIGO XML

Para cada suceso XML excepto un suceso EXCEPTION , el analizador establece el valor del registro especial XML-CODE en cero. Para un suceso EXCEPTION , el analizador establece XML-CODE en un valor que identifica la excepción específica.

Para obtener información sobre los posibles códigos de excepción, consulte las referencias relacionadas.

Cuando el analizador devuelve el control a la sentencia XML PARSE desde el procedimiento de proceso, XML-CODE generalmente contiene el valor más reciente establecido por el analizador. Sin embargo, para cualquier suceso que no sea EXCEPTION, si establece XML-CODE en -1 en el procedimiento de proceso, el análisis termina con una condición de excepción iniciada por el usuario cuando el control vuelve al analizador, y XML-CODE conserva el valor -1.

Para un EXCEPTION suceso XML, el procedimiento de proceso puede, en algunos casos, establezca XML-CODE en un valor significativo antes de que el control vuelva al analizador. (Para obtener detalles, consulte las tareas relacionadas sobre el manejo de excepciones XML PARSE y el manejo de conflictos de codificación.) Si establece XML-CODE en cualquier otro valor distinto de cero o lo establece para cualquier otra excepción, el analizador restablece XML-CODE en el código de excepción original.

La tabla siguiente muestra los resultados de establecer XML-CODE en varios valores. La columna más a la izquierda muestra el tipo de suceso XML pasado al procedimiento de proceso; las otras cabeceras de columna muestran el valor XML-CODE establecido por el procedimiento de proceso. La celda en la intersección de cada fila y columna muestra la acción que el analizador realiza tras la devolución del procedimiento de proceso para una combinación determinada de suceso XML y valor XML-CODE .

Tabla 39. ***Resultados de cambios de proceso-procedimiento en XML-CODE***

Tipo de suceso XML	**-1**	**0**	**XML-CODE-100,000 (EBCDIC) XML-CODE-200,000 (ASCII)**	**Otro valor distinto de cero**
Excepción de conflicto de codificación (códigos de excepción 50-99)	Ignora el valor; mantiene el valor XML-CODE original	Elige la codificación en función del código de excepción específico[1]	Ignora el valor; mantiene el valor XML-CODE original	Ignora el valor; mantiene el valor XML-CODE original
Excepción de opción de codificación (códigos de excepción > 100.000)	Ignora el valor; mantiene el valor XML-CODE original	Análisis utilizando el valor de página de códigos externa[2]	Analiza utilizando la diferencia (mostrada anteriormente) como el valor de codificación[2]	Ignora el valor; mantiene el valor XML-CODE original
Otra excepción	Ignora el valor; mantiene el valor XML-CODE original	Continuación limitada sólo para los códigos de excepción 1-49[3]	Ignora el valor; mantiene el valor XML-CODE original	Ignora el valor; mantiene el valor XML-CODE original
Suceso normal	Finaliza inmediatamente; XML-CODE = -1[4]	[Sin cambio aparente en XML-CODE]	Finaliza inmediatamente; XML-CODE = -1	Finaliza inmediatamente; XML-CODE = -1

1. Consulte los códigos de excepción en la referencia relacionada sobre XML PARSE excepciones.
2. Consulte la tarea relacionada sobre el manejo de conflictos de codificación.
3. Consulte la tarea relacionada sobre el manejo de excepciones XML PARSE .
4. Consulte la tarea relacionada sobre la terminación del análisis XML.

La generación XML también utiliza el registro especial XML-CODE . Para obtener detalles, consulte la tarea relacionada sobre el manejo de excepciones XML GENERATE .

Conceptos relacionados
"Cómo maneja el analizador XML los errores" en la página 431

Tareas relacionadas
"Escritura de procedimientos para procesar XML" en la página 422
"Manejo de excepciones XML PARSE" en la página 430
"Manejo de conflictos de codificación" en la página 432
"Terminando análisis XML" en la página 433
"Manejo de excepciones XML GENERATE" en la página 445

Referencias relacionadas
"Excepciones XML PARSE" en la página 597
"excepciones XML GENERATE" en la página 608
XML-CODE (*COBOL for Linux en x86 Consulta de lenguaje*)
XML-EVENT (*COBOL for Linux en x86 Consulta de lenguaje*)

XML-TEXT y XML-NTEXT

Para la mayoría de sucesos XML, el analizador establece XML-TEXT o XML-NTEXT en un fragmento de documento asociado.

Normalmente, el analizador establece XML-TEXT si el documento XML está en un elemento de datos alfanumérico. El analizador establece XML-NTEXT si:

- El documento XML está en un elemento de datos nacional.
- El documento XML está en un elemento de datos alfanumérico y se produce el suceso ATTRIBUTE-NATIONAL-CHARACTER o CONTENT-NATIONAL-CHARACTER .

Los registros especiales XML-TEXT y XML-NTEXT son mutuamente excluyentes. Cuando el analizador establece XML-TEXT, XML-NTEXT está vacío con la longitud cero. Cuando el analizador establece XML-NTEXT, XML-TEXT está vacío con la longitud cero.

Para determinar el número de unidades de codificación de caracteres en XML-NTEXT, utilice la función intrínseca LENGTH ; por ejemplo, FUNCTION LENGTH(XML-NTEXT). Para determinar el número de bytes en XML-NTEXT, utilice el registro especial LENGTH OF XML-NTEXT. El número de unidades de codificación de caracteres difiere del número de bytes.

Para determinar el número de bytes en XML-TEXT, utilice el registro especial LENGTH OF XML-TEXT o la función intrínseca LENGTH ; cada uno devuelve el número de bytes.

Los registros especiales XML-TEXT y XML-NTEXT no están definidos fuera del procedimiento de proceso.

Conceptos relacionados
"Sucesos XML" en la página 423
"CÓDIGO XML" en la página 424

Tareas relacionadas
"Escritura de procedimientos para procesar XML" en la página 422

Referencias relacionadas
XML-TEXT (*COBOL for Linux en x86 Consulta de lenguaje*)
XML-NTEXT (*COBOL for Linux en x86 Consulta de lenguaje*)

Transformación de texto XML en elementos de datos COBOL

Acerca de esta tarea

Puesto que los datos XML no son de longitud fija ni de formato fijo, es necesario utilizar técnicas especiales al mover datos XML a un elemento de datos COBOL.

Para los elementos alfanuméricos, decida si los datos XML deben ir al final izquierdo (predeterminado), o al final derecho, del elemento de datos COBOL. Si los datos deben ir en el extremo derecho, especifique la cláusula `JUSTIFIED RIGHT` en la definición del elemento.

Tenga en cuenta de forma especial los valores XML numéricos, en particular los valores monetarios "decorados" como '$1,234.00' o '$1234'. Estas dos series pueden significar lo mismo en XML, pero necesitan definiciones muy diferentes si se utilizan como campos de envío COBOL.

Utilice una de las técnicas siguientes cuando mueva datos XML a elementos de datos COBOL:

- Si el formato es razonablemente regular, codifique un `MOVE` en un elemento alfanumérico que redefina adecuadamente como un elemento editado numéricamente. A continuación, realice el movimiento final a un elemento numérico (operativo) moviendo desde, y por lo tanto deseditando, el elemento editado numérico. (Un formato regular tendría el mismo número de dígitos después de la coma decimal, un separador de coma para valores mayores que 999, y así sucesivamente.)
- Para simplificar y aumentar enormemente la flexibilidad, utilice las siguientes funciones intrínsecas para datos XML alfanuméricos:
  - `NUMVAL` para extraer y decodificar valores numéricos simples de datos XML que representan números sin formato
  - `NUMVAL-C` para extraer y decodificar valores numéricos de datos XML que representan cantidades monetarias

  Sin embargo, el uso de estas funciones se realiza a expensas del rendimiento.

Tareas relacionadas
“Conversión a números (NUMVAL, NUMVAL-C)” en la página 107
“Utilización de datos nacionales (Unicode) en COBOL” en la página 191
“Escritura de procedimientos para procesar XML” en la página 422

La codificación de documentos XML

Los documentos XML deben codificarse en una página de códigos soportada.

Los documentos XML que analice utilizando sentencias `XML PARSE` deben codificarse y los documentos XML que cree utilizando sentencias `XML GENERATE` deben codificarse de la forma siguiente:

- Documentos en elementos de datos nacionales: en formato Unicode UTF-16 en little-endian
- Documentos en elementos de datos alfanuméricos nativos: en Unicode UTF-8 o una página de códigos ASCII de un solo byte soportada por las bibliotecas de conversión ICU (International Components for Unicode)

  Un *elemento de datos alfanuméricos nativos* es un elemento de datos alfanuméricos de categoría que se compila con la opción de compilador `CHAR(NATIVE)` en vigor o cuya entrada de descripción de datos contiene la frase `NATIVE` .
- Documentos en elementos de datos alfanuméricos de host: en una página de códigos EBCDIC de un solo byte soportada por las bibliotecas de conversión ICU

  Un *elemento de datos alfanuméricos de host* es un elemento de datos alfanuméricos de categoría que se compila con la opción de compilador `CHAR(EBCDIC)` en vigor y cuya entrada de descripción de datos no contiene la frase `NATIVE` .

Las codificaciones soportadas por las bibliotecas de conversión de ICU se documentan en la referencia relacionada sobre el explorador de conversores de ICU.

Conceptos relacionados
“Codificación de documento de entrada XML” en la página 427

Tareas relacionadas
“Especificación de la codificación” en la página 428
“Análisis de documentos XML codificados en UTF-8” en la página 429
Capítulo 20, “Producción de salida XML”, en la página 439

Referencias relacionadas
“char” en la página 272
Componentes internacionales para Unicode: Explorador de conversores

Codificación de documento de entrada XML

Para analizar un documento XML utilizando la sentencia XML PARSE , el documento debe estar codificado en una codificación soportada.

Las codificaciones soportadas para una operación de análisis determinada dependen del tipo del elemento de datos que contiene el documento XML. El analizador da soporte a los siguientes tipos de elementos de datos y codificaciones:

- Category elementos de datos nacionales con contenido codificado en Unicode UTF-16 en formato little-endian
- Elementos de datos alfanuméricos nativos con contenido codificado en Unicode UTF-8 o una de las páginas de códigos ASCII de un solo byte soportadas
- Elementos de datos alfanuméricos de host con contenido codificado en una de las páginas de códigos EBCDIC de un solo byte soportadas

Las páginas de códigos soportadas se describen en la referencia relacionada sobre la codificación de documentos XML.

El analizador determina la *codificación de documento real* examinando los primeros bytes del documento XML. Si la codificación de documento real es ASCII o EBCDIC, el analizador necesita información de página de códigos específica para poder analizar correctamente. Esta información de página de códigos adicional se obtiene de la declaración de codificación de documento o de la información de página de códigos externa.

La declaración de codificación de documento es una parte opcional de la declaración XML al principio del documento. Para obtener detalles, consulte la tarea relacionada sobre cómo especificar la codificación.

La *página de códigos externa* para documentos XML ASCII (la *página de códigos ASCII externa*) es la página de códigos indicada por el entorno local de ejecución actual. La página de códigos externa para documentos XML EBCDIC (la *página de códigos EBCDIC externa*) es una de estas:

- La página de códigos que ha especificado en la variable de entorno EBCDIC_CODEPAGE
- La página de códigos EBCDIC predeterminada seleccionada para el entorno local de tiempo de ejecución actual si no ha establecido la variable de entorno EBCDIC_CODEPAGE

Si la codificación especificada no es uno de los juegos de caracteres codificados soportados, el analizador señala un suceso de excepción XML antes de iniciar la operación de análisis. Si la codificación de documento real no coincide con la codificación especificada, el analizador señala una excepción XML adecuada después de iniciar la operación de análisis.

Para analizar un documento XML codificado en una página de códigos no soportada, primero convierta el documento en datos de caracteres nacionales (UTF-16) utilizando la función intrínseca NATIONAL-OF . Puede convertir los fragmentos individuales de texto de documento que se pasan al procedimiento de proceso en el registro especial XML-NTEXT de nuevo a la página de códigos original utilizando la función intrínseca DISPLAY-OF .

Declaración XML y espacio en blanco:

Los documentos XML pueden empezar por *espacio en blanco* sólo si no tienen una declaración XML:

- Si un documento XML empieza con una declaración XML, el primer corchete angular (<) en el documento debe ser el primer carácter del documento.
- Si un documento XML no empieza con una declaración XML, el primer corchete angular del documento sólo puede ir precedido de un espacio en blanco.

Los caracteres de espacio en blanco tienen los valores hexadecimales que se muestran en la tabla siguiente.

Tabla 40. ***Valores hexadecimales de caracteres de espacio en blanco***		
Carácter de espacio en blanco	**EBCDIC**	**Unicode/ASCII**
Espacio	X'40 "	X'20 "
Tabulación horizontal	X'05 "	X'09 "
Retorno de carro	X'0D'	X'0D'
Salto de línea	X'25 "	X'0A'
Línea nueva/línea siguiente	X'15 "	X'85 "

Tareas relacionadas
“Conversión a o desde representación nacional (Unicode)” en la página 199
“Especificación de la codificación” en la página 428
“Análisis de documentos XML codificados en UTF-8” en la página 429
“Manejo de excepciones XML PARSE” en la página 430

Referencias relacionadas
“Entornos locales y páginas de códigos soportadas” en la página 217
“La codificación de documentos XML” en la página 426
“Caracteres sensibles a la página de códigos EBCDIC en la marcación XML” en la página 429
“Excepciones XML PARSE” en la página 597

Especificación de la codificación

Puede elegir cómo especificar la codificación para analizar un documento XML que se encuentra en un elemento de datos alfanumérico.

Acerca de esta tarea

La forma preferida es omitir la declaración de codificación del documento y basarse en la especificación de página de códigos externa.

La omisión de la declaración de codificación permite transmitir más fácilmente un documento XML entre sistemas heterogéneos. (Si ha incluido una declaración de codificación, deberá actualizarla para reflejar cualquier conversión de página de códigos impuesta por el proceso de transmisión.)

La página de códigos utilizada para analizar un documento XML alfanumérico que no tiene una declaración de codificación es la página de códigos de tiempo de ejecución.

En su lugar, puede especificar una declaración de codificación en la declaración XML con la que empiezan la mayoría de los documentos XML. Por ejemplo:

```
<?xml version="1.0" encoding="ibm-1140"?>
```

Tenga en cuenta que el analizador XML genera una excepción si encuentra una declaración XML que no empieza en el primer byte de un documento XML.

Si especifica una declaración de codificación, utilice uno de los nombres de página de códigos primarios o alias soportados por las bibliotecas de conversión ICU. Los nombres de página de códigos se documentan en la referencia relacionada sobre el explorador de conversores de ICU.

Para obtener más información sobre los CCSID soportados para el análisis XML, consulte la referencia relacionada sobre la codificación de documentos XML.

Conceptos relacionados
“Codificación de documento de entrada XML” en la página 427

Tareas relacionadas
“Análisis de documentos XML codificados en UTF-8” en la página 429
“Manejo de conflictos de codificación” en la página 432

Referencias relacionadas
“Entornos locales y páginas de códigos soportadas” en la página 217
“La codificación de documentos XML” en la página 426
Componentes internacionales para Unicode: Explorador de conversores

Caracteres sensibles a la página de códigos EBCDIC en la marcación XML

Varios caracteres especiales que se utilizan en la marcación XML tienen diferentes representaciones hexadecimales en diferentes páginas de códigos EBCDIC.

La tabla siguiente muestra estos caracteres especiales y sus valores hexadecimales para varios CCSID EBCDIC.

Tabla 41. ***Valores hexadecimales de caracteres especiales para varios CCSID EBCDIC***

Carácter	**1047**	**1140**	**1141**	**1142**	**1143**	**1144**	**1145**	**1146**	**1147**	**1148**	**1149**
[	X'AD "	X'BA "	X'63 "	X'9E'	X'B5'	X90	X'4A'	X'B1'	X90	X'4A'	X'AE "
]	X'BD "	X'BB '	X'FC "	X'9F'	X'9F'	X'51 "	X'5A'	X'BB '	X'B5'	X'5A'	X'9E'
!	X'5A'	X'5A'	X'4F'	X'4F'	X'4F'	X'4F'	X'BB '	X'5A'	X'4F'	X'4F'	X'4F'
|	X'4F'	X'4F'	X'BB '	X'BB '	X'BB '	X'BB '	X'4F'	X'4F'	X'BB '	X'BB '	X'BB '
#	X'7B'	X'7B'	X'7B'	X'4A'	X'63 "	X'B1'	X'69 "	X'7B'	X'B1'	X'7B'	X'7B'

Análisis de documentos XML codificados en UTF-8

Puede analizar documentos XML codificados en Unicode UTF-8 de una forma similar al análisis de otros documentos XML. Sin embargo, se aplican algunos requisitos adicionales.

Acerca de esta tarea

Para analizar un documento XML UTF-8 , codifique la sentencia XML PARSE como lo haría normalmente para analizar documentos XML:

```
XML PARSE xml-document
    PROCESSING PROCEDURE xml-event-handler
    . . .
END-XML
```

Sin embargo, tenga en cuenta los siguientes requisitos adicionales:

- El elemento de datos de análisis (`xml-document` en el ejemplo anterior) debe ser alfanumérico de categoría y la opción de compilador `CHAR(EBCDIC)` no debe estar en vigor.
- Para que el documento XML se analice como UTF-8 en lugar de ASCII, asegúrese de que se cumpla al menos una de las condiciones siguientes:
  - El entorno local de tiempo de ejecución es un entorno local UTF-8 .
  - El documento contiene una declaración de codificación XML que especifica UTF-8 (`encoding="UTF-8"`).
  - El documento empieza con una marca de orden de bytes UTF-8 .
- El documento no debe contener ningún carácter que tenga un valor escalar Unicode mayor que x 'FFFF'. Utilice una referencia de caracteres ("&#x*hhhh*;") para dichos caracteres.

El analizador devuelve los fragmentos de documento XML en los registros regístrese `XML-TEXT`.

Los caracteres UTF-8 se codifican utilizando un número variable de bytes por carácter. La mayoría de operaciones COBOL en datos alfanuméricos presuponen una codificación de un solo byte, en la que cada carácter se codifica en 1 byte. Cuando opera con caracteres UTF-8 como datos alfanuméricos, debe asegurarse de que los datos se procesen correctamente. Evite operaciones (como la modificación de referencia y los movimientos que implican el truncamiento) que pueden dividir un carácter de varios bytes entre bytes. No puede utilizar de forma fiable sentencias como INSPECT para procesar caracteres de varios bytes en datos alfanuméricos.

Conceptos relacionados
"XML-TEXT y XML-NTEXT" en la página 425

Tareas relacionadas
"Proceso de datos UTF-8 utilizando tipos de datos UTF-16 (nacional)" en la página 208
"Análisis de documentos XML" en la página 421
"Especificación de la codificación" en la página 428

Referencias relacionadas
"char" en la página 272
"La codificación de documentos XML" en la página 426
sentencia XML PARSE (*COBOL for Linux en x86 Consulta de lenguaje*)

Manejo de excepciones XML PARSE

Si el analizador XML encuentra una anomalía o un error durante el análisis, establece un código de excepción en el XML-CODE registro especial y señala un suceso de excepción XML.

Acerca de esta tarea

Si el código de excepción está dentro de un rango determinado, es posible que pueda manejar el suceso de excepción dentro del procedimiento de proceso y reanudar el análisis.

Para manejar una excepción en el procedimiento de proceso, siga estos pasos:

Procedimiento

1. Compruebe el contenido de XML-CODE.
2. Maneje la excepción correctamente.
3. Establezca XML-CODE en cero para indicar que ha manejado la excepción.
4. Devuelve el control al analizador.

Resultados

La condición de excepción ya no existe.

Puede manejar las excepciones de esta forma sólo si el código de excepción que se pasa en XML-CODE está dentro de uno de los rangos siguientes, lo que indica que se ha detectado un conflicto de codificación:

- 50 - 99
- 100,001 - 165,535
- 200,001 - 265,535

Códigos de excepción 1-49: En el procedimiento de proceso, puede realizar un manejo limitado de las excepciones para las que el código de excepción está dentro del rango de 1 a 49. Después de que se produzca una excepción en este rango, el analizador no señala más sucesos normales, excepto el suceso END-OF-DOCUMENT , incluso si establece XML-CODE en cero antes de devolver. Si establece XML-CODE en cero, el analizador continúa analizando el documento y señala las excepciones que encuentre. (Si lo hace, puede proporcionar una forma útil de descubrir varios errores en el documento.)

Restricción: Es posible que el analizador XML COBOL de no señale todos los sucesos de excepción adicionales. El número de excepciones está limitado al espacio restante en la matriz de señales de sucesos de XML PARSE , probablemente 8192 sucesos.

Al final del análisis después de una excepción que tiene un código de excepción en el rango de 1 a 49, el control se pasa a la sentencia especificada en la frase ON EXCEPTION . Si no ha codificado una frase ON EXCEPTION , el control se pasa al final de la sentencia XML PARSE . XML-CODE contiene el código establecido por el analizador para la excepción más reciente.

Para todas las excepciones distintas de las que tienen un código de excepción dentro de uno de los rangos descritos anteriormente, el analizador no señala ningún suceso adicional, pero pasa el control a la sentencia especificada en la frase ON EXCEPTION . XML-CODE contiene el código de excepción original aunque establezca XML-CODE en el procedimiento de proceso antes de devolver el control al analizador.

Si no desea manejar una excepción, devuelva el control al analizador sin cambiar el valor de XML-CODE. El analizador transfiere el control a la sentencia especificada en la frase ON EXCEPTION . Si no ha codificado una frase ON EXCEPTION , el control se transfiere al final de la sentencia XML PARSE .

Si no se producen excepciones no manejadas antes del final del análisis, el control se pasa a la sentencia especificada en la frase NOT ON EXCEPTION . Si no ha codificado una frase NOT ON EXCEPTION , el control se transfiere al final de la sentencia XML PARSE . XML-CODE contiene cero.

Conceptos relacionados
“CÓDIGO XML” en la página 424
“Codificación de documento de entrada XML” en la página 427
“Cómo maneja el analizador XML los errores” en la página 431

Tareas relacionadas
“Escritura de procedimientos para procesar XML” en la página 422
“Manejo de conflictos de codificación” en la página 432

Referencias relacionadas
“La codificación de documentos XML” en la página 426
“Excepciones XML PARSE” en la página 597

Cómo maneja el analizador XML los errores

Cuando el analizador XML detecta un error en un documento XML, genera un suceso de excepción XML y pasa el control al procedimiento de proceso.

El analizador pasa la siguiente información en registros especiales al procedimiento de proceso:

- XML-EVENT contiene 'EXCEPTION'.
- XML-CODE contiene un código de excepción numérico.

  Los códigos de excepción se describen en las referencias relacionadas sobre las excepciones XML PARSE .
- Para excepciones muy graves, XML-TEXT o XML-NTEXT contiene el texto del documento hasta el punto en el que se ha detectado la excepción, inclusive.
- Para las excepciones de aviso emitidas para utilizar un prefijo no declarado, XML-TEXT o XML-NTEXT contiene el nombre de atributo completo o el nombre de elemento. Es decir, el nombre incluye el prefijo no declarado y el separador dos puntos (:).
- XML-TEXT o XML-NTEXT contiene el texto del documento hasta el punto en el que se ha detectado la excepción, inclusive.

Todos los demás registros especiales XML están vacíos con longitud cero.

Es posible que el procedimiento de proceso pueda manejar una excepción para que el análisis continúe si el código de excepción está dentro de uno de los rangos siguientes:

- 1 - 99
- 100,001 - 165,535

- 200,001 - 265,535

Si el código de excepción tiene cualquier otro valor distinto de cero, el análisis no puede continuar.

Conflictos de codificación: las excepciones para los conflictos de codificación (50-99 y 300-399) se señalan antes de que empiece el análisis del documento. Para estas excepciones, XML-TEXT o XML-NTEXT tiene una longitud cero o sólo contiene el valor de declaración de codificación del documento.

Códigos de excepción 1-49: una excepción para la que el código de excepción está en el rango 1-49 es un error muy grave de acuerdo con la *especificación XML*. Por lo tanto, el analizador no continúa el análisis normal aunque el procedimiento de proceso maneje la excepción. Sin embargo, el analizador continúa explorando si hay más errores hasta que alcanza el final del documento, o hasta que se agota la matriz de señales EVENT XML existente. Para estas excepciones, el analizador no señala más sucesos normales excepto el suceso END-OF-DOCUMENT .

Conceptos relacionados
"Sucesos XML" en la página 423
"CÓDIGO XML" en la página 424
"Codificación de documento de entrada XML" en la página 427

Tareas relacionadas
"Manejo de excepciones XML PARSE" en la página 430
"Manejo de conflictos de codificación" en la página 432

"Terminando análisis XML" en la página 433

Referencias relacionadas
"La codificación de documentos XML" en la página 426
"Excepciones XML PARSE" en la página 597
Especificación XML

Manejo de conflictos de codificación

Acerca de esta tarea

Es posible que el procedimiento de proceso pueda manejar excepciones para conflictos de codificación de documentos. Sucesos de excepción en los que el elemento de datos de análisis es alfanumérico y el código de excepción de XML-CODE está dentro del rango de 100.001 a 165.535 o 200.001 a 265.535 indican que la página de códigos del documento (tal como se especifica mediante su declaración de codificación) entra en conflicto con la información de página de códigos externa.

En este caso especial, puede optar por analizar utilizando la página de códigos del documento restando 100.000 o 200.000 del valor de XML-CODE (en función de si la página de códigos es EBCDIC o ASCII, respectivamente). Por ejemplo, si XML-CODE contiene 101.140, la página de códigos del documento es 1140. Como alternativa, puede elegir analizar utilizando la *página de códigos externa* estableciendo XML-CODE en cero antes de volver al analizador.

El analizador realiza una de las tres acciones siguientes después de volver de un procedimiento de proceso para un suceso de excepción de conflicto de codificación:

- Si establece XML-CODE en cero, el analizador utiliza la página de códigos ASCII externa de o la página de códigos EBCDIC externa, en función de si el elemento de datos de análisis es un elemento alfanumérico nativo o un elemento alfanumérico de host, respectivamente.
- Si establece XML-CODE en la página de códigos del documento (es decir, el valor XML-CODE original menos 100.000 o 200.000, según corresponda), el analizador utiliza la página de códigos del documento.

  Este es el único caso en el que el analizador continúa cuando XML-CODE tiene un valor distinto de cero al volver de un procedimiento de proceso.

- De lo contrario, el analizador deja de procesar el documento y devuelve el control a la sentencia XML PARSE con una condición de excepción. XML-CODE contiene el código de excepción que se ha pasado originalmente con el suceso de excepción.

Conceptos relacionados
"CÓDIGO XML" en la página 424
"Codificación de documento de entrada XML" en la página 427
"Cómo maneja el analizador XML los errores" en la página 431

Tareas relacionadas
"Manejo de excepciones XML PARSE" en la página 430

Referencias relacionadas
"La codificación de documentos XML" en la página 426
"Excepciones XML PARSE" en la página 597

Terminando análisis XML

Puede terminar el análisis inmediatamente, sin procesar ningún texto XML restante, estableciendo XML-CODE en -1 en el procedimiento de proceso antes de que el procedimiento vuelva al analizador desde cualquier suceso XML normal (es decir, cualquier suceso que no sea EXCEPTION).

Acerca de esta tarea

Puede utilizar esta técnica cuando el procedimiento de proceso ha examinado suficiente del documento o ha detectado alguna irregularidad en el documento que impide un proceso más significativo.

Si termina el análisis de esta forma, el analizador no señala ningún suceso XML adicional, incluido el suceso de excepción. El control se transfiere a la frase ON EXCEPTION de la sentencia XML PARSE , si se ha especificado dicha frase.

En la sentencia imperativa de la frase ON EXCEPTION , puede determinar si el análisis se ha terminado deliberadamente comprobando si XML-CODE contiene -1. Si no especifica la frase ON EXCEPTION , el control se transfiere al final de la sentencia XML PARSE .

También puede terminar el análisis después de cualquier suceso XML EXCEPTION volviendo al analizador desde el procedimiento de proceso sin cambiar el valor en XML-CODE. El resultado es similar al resultado de la terminación deliberada, excepto que el analizador vuelve a la sentencia XML PARSE con XML-CODE que contiene el código de excepción original.

Conceptos relacionados
"CÓDIGO XML" en la página 424
"Cómo maneja el analizador XML los errores" en la página 431

Tareas relacionadas
"Escritura de procedimientos para procesar XML" en la página 422
"Manejo de excepciones XML PARSE" en la página 430

Ejemplos de XML PARSE

Los ejemplos a los que se hace referencia a continuación ilustran diversos usos de la sentencia XML PARSE .

"Ejemplo: análisis de un documento simple" en la página 434
"Ejemplo: programa para procesar XML" en la página 434

Ejemplo: análisis de un documento simple

Este ejemplo muestra el flujo de sucesos y el contenido del registro especial XML-TEXT que resulta del análisis de un documento XML simple.

Supongamos que el programa COBOL contiene el siguiente documento XML en el elemento de datos Doc:

```
<?xml version="1.0"?><msg type="short">Hello, World!</msg>
```

El fragmento de código siguiente muestra una sentencia XML PARSE para analizar Docy un procedimiento de proceso, P, para manejar los sucesos XML:

```
XML Parse Doc
   Processing procedure P
   . . .
P. Display XML-Event XML-Text.
```

El procedimiento de proceso muestra el contenido de XML-EVENT y XML-TEXT para cada suceso que el analizador señala durante el análisis. La tabla siguiente muestra los sucesos y el texto.

Tabla 42. ***Sucesos XML y registros especiales***

XML-EVENT	**XML-TEXT**
START-OF-DOCUMENT	
VERSION-INFORMATION	1.0
START-OF-ELEMENT	msg
ATTRIBUTE-NAME	type
ATTRIBUTE-CHARACTERS	short
CONTENT-CHARACTERS	Hello, World!
END-OF-ELEMENT	msg
END-OF-DOCUMENT	

Conceptos relacionados
“Sucesos XML” en la página 423
“XML-TEXT y XML-NTEXT” en la página 425

Ejemplo: programa para procesar XML

Este ejemplo muestra el análisis de un documento XML y un procedimiento de proceso que informa de los distintos sucesos XML y sus fragmentos de texto asociados.

El documento XML se muestra en el origen del programa para facilitar el seguimiento del flujo del análisis. La salida del programa se muestra después del ejemplo.

Para comprender la interacción del analizador y el procedimiento de proceso, y para comparar sucesos con fragmentos de documento, compare el documento XML con la salida del programa.

```
Process codepage(1047)
       Identification division.
         Program-id. XMLSAMPL.
       Data division.
        Working-storage section.
      ****************************************************************
      * XML document data, encoded as initial values of data items.    *
      ****************************************************************
         1 xml-document-data.
          2 pic x(39) value '<?xml version="1.0" encoding="UTF-8"'.
          2 pic x(19) value ' standalone="yes"?>'.
          2 pic x(39) value '<!--This document is just an example-->'.
```

```
    2 pic x(10) value '<sandwich>'.
    2 pic x(33) value '<bread type="baker&apos;s best"/>'.
    2 pic x(36) value '<?spread We'll use real mayonnaise?>'.
    2 pic x(29) value '<meat>Ham &amp; turkey</meat>'.
    2 pic x(34) value '<filling>Cheese, lettuce, tomato, '.
    2 pic x(32) value 'and that's all, Folks!</filling>'.
    2 pic x(25) value '<![CDATA[We should add a '.
    2 pic x(20) value '<relish> element!]]>'.
    2 pic x(28) value '<listprice>$4.99</listprice>'.
    2 pic x(25) value '<discount>0.10</discount>'.
    2 pic x(31) value '</sandwich>'.
*****************************************************************
* XML document, represented as fixed-length records.            *
*****************************************************************
   1 xml-document redefines xml-document-data.
    2 xml-segment pic x(40) occurs 10 times.
   1 xml-segment-no comp pic s9(4).
   1 content-buffer pic x(100).
   1 current-element-stack.
    2 current-element pic x(30) occurs 10 times.
*****************************************************************
* Sample data definitions for processing numeric XML content.    *
*****************************************************************
   1 element-depth comp pic s9(4).
   1 discount computational pic 9v99 value 0.
   1 display-price pic $$9.99.
   1 filling pic x(4095).
   1 list-price computational pic 9v99 value 0.
   1 ofr-ed pic x(9) justified.
   1 ofr-ed-1 redefines ofr-ed pic 999999.99.
 Procedure division.
  Mainline section.
     Move 1 to xml-segment-no
     Display 'Initial segment {' xml-segment(xml-segment-no) '}'
     Display ' '
     XML parse xml-segment(xml-segment-no)
         processing procedure XML-handler
       On exception
         Display 'XML processing error, XML-Code=' XML-Code '.'
         Move 16 to return-code
         Goback
       Not on exception
         Display 'XML document successfully parsed.'
     End-XML
*****************************************************************
* Process the transformed content and calculate promo price.     *
*****************************************************************
     Display ' '
     Display '-----++++++++++ Using information from XML '
         '**********-----'
     Display ' '
     Move list-price to Display-price
     Display ' Sandwich list price: ' Display-price
     Compute Display-price = list-price * (1 - discount)
     Display ' Promotional price: ' Display-price
     Display ' Get one today!'
     Goback.
  XML-handler section.
     Evaluate XML-Event
* ==> Order XML events most frequent first
       When 'START-OF-ELEMENT'
         Display 'Start element tag: {' XML-Text '}'
         Add 1 to element-depth
         Move XML-Text to current-element(element-depth)
       When 'CONTENT-CHARACTERS'
         Display 'Content characters: {' XML-Text '}'
* ==> In general, a split can occur for any element or attribute
* ==> data, but in this sample, it only occurs for "filling"...
         If xml-information = 2 and
             current-element(element-depth) not = 'filling'
           Display 'Unexpected split in content for element '
               current-element(element-depth)
           Move -1 to xml-code
         End-if
* ==> Transform XML content to operational COBOL data item...
         Evaluate current-element(element-depth)
           When 'filling'
* ==> After reassembling separate pieces of character content...
             String xml-text delimited by size into
                 content-buffer with pointer tally
               On overflow
                 Display 'content buffer ('
```

```
                        length of content-buffer
                        ' bytes) is too small'
                    Move -1 to xml-code
                End-string
                Evaluate xml-information
                  When 2
                    Display '  Character data for element "filling" '
                        'is incomplete.'
                    Display '  The partial data was buffered for '
                        'content assembly.'
                  When 1
                    subtract 1 from tally
                    move content-buffer(1:tally) to filling
                    Display '  Element "filling" data (' tally
                        ' bytes) is now complete:'
                    Display '  {' filling(1:tally) '}'
                End-evaluate
              When 'listprice'
      * ==> Using function NUMVAL-C...
                Move XML-Text to content-buffer
                Compute list-price =
                    function numval-c(content-buffer)
              When 'discount'
      * ==> Using de-editing of a numeric edited item...
                Move XML-Text to ofr-ed
                Move ofr-ed-1 to discount
            End-evaluate
          When 'END-OF-ELEMENT'
            Display 'End element tag: {' XML-Text '}'
            Subtract 1 from element-depth
          When 'END-OF-INPUT'
            Display 'End of input'
            Add 1 to xml-segment-no
            Display '  Next segment: {' xml-segment(xml-segment-no)
                '}'
            Display ' '
            Move 1 to xml-code
          When 'START-OF-DOCUMENT'
            Display 'Start of document'
            Move 0 to element-depth
            Move 1 to tally
          When 'END-OF-DOCUMENT'
            Display 'End of document.'
          When 'VERSION-INFORMATION'
            Display 'Version: {' XML-Text '}'
          When 'ENCODING-DECLARATION'
            Display 'Encoding: {' XML-Text '}'
          When 'STANDALONE-DECLARATION'
            Display 'Standalone: {' XML-Text '}'
          When 'ATTRIBUTE-NAME'
            Display 'Attribute name: {' XML-Text '}'
          When 'ATTRIBUTE-CHARACTERS'
            Display 'Attribute value characters: {' XML-Text '}'
          When 'ATTRIBUTE-CHARACTER'
            Display 'Attribute value character: {' XML-Text '}'
          When 'START-OF-CDATA-SECTION'
            Display 'Start of CData section'
          When 'END-OF-CDATA-SECTION'
            Display 'End of CData section'
          When 'CONTENT-CHARACTER'
            Display 'Content character: {' XML-Text '}'
          When 'PROCESSING-INSTRUCTION-TARGET'
            Display 'PI target: {' XML-Text '}'
          When 'PROCESSING-INSTRUCTION-DATA'
            Display 'PI data: {' XML-Text '}'
          When 'COMMENT'
            Display 'Comment: {' XML-Text '}'
          When 'EXCEPTION'
            Compute tally = function length (XML-Text)
            Display 'Exception ' XML-Code ' at offset ' tally '.'
          When other
            Display 'Unexpected XML event: ' XML-Event '.'
        End-evaluate
        .
       End program XMLSAMPL.
```

Salida del análisis de

En la salida siguiente puede ver qué fragmentos del documento se han asociado con los sucesos que se han producido durante el análisis:

```
Start of document
Version: {1.0}
Encoding: {UTF-8}
Standalone: {yes}
Comment: {This document is just an example}
Start element tag: {sandwich}
Content characters: {  }
Start element tag: {bread}
Attribute name: {type}
Attribute value characters: {baker}
Attribute value character: {'}
Attribute value characters: {s best}
End element tag: {bread}
Content characters: {  }
PI target: {spread}
PI data: {please use real mayonnaise  }
Content characters: {  }
Start element tag: {meat}
Content characters: {Ham }
Content character: {&}
Content characters: { turkey}
End element tag: {meat}
Content characters: {  }
Start element tag: {filling}
Content characters: {Cheese, lettuce, tomato, etc.}
End element tag: {filling}
Content characters: {  }
Start of CData: {<![CDATA[}
Content characters: {We should add a <relish> element in future!}
End of CData: {]]>}
Content characters: {  }
Start element tag: {listprice}
Content characters: {$4.99 }
End element tag: {listprice}
Content characters: {  }
Start element tag: {discount}
Content characters: {0.10}
End element tag: {discount}
End element tag: {sandwich}
End of document.
XML document successfully parsed

-----++++++++++ Using information from XML **********-----

  Sandwich list price:  $4.99
  Promotional price:    $4.49
  Get one today!
```

Conceptos relacionados

“Sucesos XML” en la página 423

Referencias relacionadas

XML-EVENT (*COBOL for Linux en x86 Consulta de lenguaje*)

Capítulo 20. Producción de salida XML

Puede producir salida XML a partir de un programa COBOL utilizando la sentencia XML GENERATE .

Acerca de esta tarea

En la sentencia XML GENERATE , identifique el origen y los elementos de datos de salida. Opcionalmente, también puede identificar:

- Un campo para recibir un recuento de los caracteres XML generados
- La codificación del documento XML generado
- Un *espacio de nombres* para el documento generado
- Un prefijo de espacio de nombres para calificar el código de inicio y final de cada elemento, si especifica un espacio de nombres
- Un elemento definido por el usuario o un nombre de atributo en el documento XML generado
- Atributos o elementos que se van a suprimir de acuerdo con algunas condiciones especificadas
- Elementos concretos que se deben especificar como atributos, elementos o contenido en la salida XML generada.
- Una sentencia para recibir el control si se produce una excepción

Opcionalmente, puede generar una declaración XML para el documento y puede hacer que los elementos de datos de origen elegibles se expresen como atributos en la salida en lugar de como elementos.

Puede utilizar el registro especial XML-CODE para determinar el estado de la generación XML.

Después de transformar elementos de datos COBOL en XML, puede utilizar la salida XML resultante de varias formas, como desplegarla en un servicio web, grabándolo en un archivo, o pasándolo como parámetro a otro programa.

Tareas relacionadas
“Generando salida XML” en la página 439
“Control de la codificación de la salida XML generada” en la página 444
“Manejo de excepciones XML GENERATE” en la página 445
“Mejora de la salida XML” en la página 449

Referencias relacionadas
XML (Extensible Markup Language)
sentencia XML GENERATE (*COBOL for Linux en x86 Consulta de lenguaje*)

Generando salida XML

Para transformar datos COBOL a XML, utilice la sentencia XML GENERATE como en el ejemplo siguiente.

Acerca de esta tarea

```
XML GENERATE XML-OUTPUT FROM SOURCE-REC
       COUNT IN XML-CHAR-COUNT
   ON EXCEPTION
       DISPLAY 'XML generation error ' XML-CODE
       STOP RUN
   NOT ON EXCEPTION
       DISPLAY 'XML document was successfully generated.'
END-XML
```

En la sentencia XML GENERATE , primero debe identificar el elemento de datos (XML-OUTPUT en el ejemplo anterior) que va a recibir la salida XML. Defina el elemento de datos para que sea lo suficientemente grande para contener la salida XML generada, normalmente de cinco a 10 veces el

tamaño de los datos de origen COBOL en función de la longitud de su nombre de datos o nombres de datos.

En DATA DIVISION, puede definir el identificador de recepción como alfanumérico (ya sea un elemento de grupo alfanumérico o un elemento elemental de categoría alfanumérica) o como nacional (ya sea un elemento de grupo nacional o un elemento elemental de categoría nacional).

A continuación, identifica el elemento de datos de origen que se va a transformar al formato XML (SOURCE-REC en el ejemplo). El elemento de datos de origen puede ser un elemento de grupo alfanumérico, un elemento de grupo nacional o un elemento de datos elemental de clase alfanumérica o nacional.

Algunos elementos de datos COBOL no se transforman en XML, pero se ignoran. Los elementos de datos subordinados de un elemento de grupo alfanumérico o un elemento de grupo nacional que transforme en XML se ignoran si:

- Especifique la cláusula REDEFINES o esté subordinada a un elemento de redefinición de este tipo
- Especifique la cláusula RENAMES

Estos elementos en el elemento de datos de origen también se ignoran al generar XML:

- Elementos de datos elementales de FILLER (o sin nombre)
- Bytes de holgura insertados para elementos de datos de SYNCHRONIZED

No se inserta ningún espacio en blanco adicional (por ejemplo, nuevas líneas o sangría) para que el XML generado sea más legible.

Opcionalmente, puede codificar la frase COUNT IN para obtener el número de unidades de codificación de caracteres XML que se rellenan durante la generación de la salida XML. Si el identificador de recepción tiene una categoría nacional, el recuento está en unidades de codificación de caracteres UTF-16 . Para todas las demás codificaciones (incluyendo UTF-8), el recuento está en bytes.

Puede utilizar el campo de recuento como una longitud de modificación de referencia para obtener sólo la parte del elemento de datos de recepción que contiene la salida XML generada. Por ejemplo, XML-OUTPUT(1:XML-CHAR-COUNT) hace referencia a las primeras XML-CHAR-COUNT posiciones de caracteres de XML-OUTPUT.

Considere el siguiente fragmento de programa:

```
01  doc pic x(512).
01  docSize pic 9(9) binary.
01  G.
    05  A pic x(3) value "aaa".
    05  B.
        10  C pic x(3) value "ccc".
        10  D pic x(3) value "ddd".
    05  E pic x(3) value "eee".
    . . .
    XML Generate Doc from G
```

El código anterior genera el siguiente documento XML, en el que A, By E se expresan como elementos hijo del elemento G, y C y D se convierten en elementos hijo del elemento B:

```
<G><A>aaa</A><B><C>ccc</C><D>ddd</D></B><E>eee</E></G>
```

De forma alternativa, puede especificar la frase ATTRIBUTES de la sentencia XML GENERATE . La frase ATTRIBUTES hace que cada elemento de datos elegible incluido en el documento XML generado se exprese como un atributo del elemento XML que lo contiene, en lugar de como un elemento hijo del elemento XML que lo contiene. Para ser elegible, el elemento de datos debe ser elemental, debe tener un nombre distinto de FILLERy no debe tener una cláusula OCCURS en su entrada de descripción de datos. El elemento XML que lo contiene corresponde al elemento de datos de grupo que se superordena inmediatamente al elemento de datos elemental. Opcionalmente, puede especificar un control más preciso de qué elementos de datos deben expresarse como atributos o elementos utilizando la frase TYPE OF .

Por ejemplo, supongamos que la sentencia XML GENERATE del extracto de programa anterior se ha codificado como se indica a continuación:

```
XML Generate Doc from G with attributes
```

A continuación, el código generaría el siguiente documento XML, en el que A y E se expresan como atributos del elemento G, y C y D se convierten en atributos del elemento B:

```
<G A="aaa" E="eee"><B C="ccc" D="ddd"></B></G>
```

Opcionalmente, puede codificar la frase ENCODING de la sentencia XML GENERATE para especificar lacodificación del documento XML generado. Si no utiliza la frase ENCODING , la codificación del documento viene determinada por la categoría del elemento de datos de recepción. Para obtener más detalles, consulte la tarea relacionada siguiente sobre el control de la codificación de la salida XML generada.

Opcionalmente, puede codificar la frase XML-DECLARATION para hacer que el documento XML generado tenga una declaración XML que incluya información de versión y una declaración de codificación. Si el elemento de datos de recepción es de categoría:

- Nacional: La declaración de codificación tiene el valor UTF-16 (encoding="UTF-16").
- Alfanumérico: la declaración de codificación se deriva de la frase ENCODING , si se especifica, o de la opción de compilador entorno local de tiempo de ejecución o variable de entorno EBCDIC_CODEPAGE si no se especifica la frase ENCODING .

Por ejemplo, el fragmento de programa siguiente especifica la frase XML-DECLARATION de XML GENERATEy especifica la codificación en UTF-8:

```
01  Greeting.
    05 msg  pic x(80)  value 'Hello, world!'.
    . . .
    XML Generate Doc from Greeting
        with Encoding "UTF-8"
        with XML-declaration
    End-XML
```

El código anterior genera el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?><Greeting><msg>Hello, world!</msg></Greeting>
```

Si no codifica la frase XML-DECLARATION , no se genera una declaración XML.

Opcionalmente, puede codificar la frase NAMESPACE para especificar un *namespace* para el documento XML generado. El valor de espacio de nombres debe ser un *URI (Uniform Resource Identifier)*válido, por ejemplo, un URL (Uniform Resource Locator); para obtener más detalles, consulte el concepto relacionado sobre la sintaxis de URI a continuación.

Especifique el espacio de nombres en un identificador o literal de categoría nacional o alfanumérico.

Si especifica un espacio de nombres, pero no especifica un prefijo de espacio de nombres (descrito a continuación), el espacio de nombres se convierte en el *espacio de nombres predeterminado* para el documento. Es decir, el espacio de nombres define en el elemento raíz se aplica de forma predeterminada a cada nombre de elemento del documento, incluido el elemento raíz.

Por ejemplo, considere las siguientes definiciones de datos y sentencia XML GENERATE :

```
01  Greeting.
    05  msg  pic x(80)  value 'Hello, world!'.
01  NS  pic x(20)   value 'http://example'.
    . . .
    XML Generate Doc from Greeting
        namespace is NS
```

El documento XML resultante tiene un espacio de nombres predeterminado (`http://example`), tal como se indica a continuación:

```
<Greeting xmlns="http://example"><msg>Hello, world!</msg></Greeting>
```

Si no especifica un espacio de nombres, los nombres de elemento del documento XML generado no están en ningún espacio de nombres.

Opcionalmente, puede codificar la frase `NAMESPACE-PREFIX` para especificar un prefijo que se aplicará a la etiqueta de inicio y finalización de cada elemento del documento generado. Sólo puede especificar un prefijo si ha especificado un espacio de nombres tal como se ha descrito anteriormente.

Cuando se ejecuta la sentencia `XML GENERATE` , el valor de prefijo debe ser un nombre XML válido, pero sin dos puntos (`:`); consulte la referencia relacionada a continuación sobre los espacios de nombres para obtener más detalles. El valor puede tener espacios finales, que se eliminan antes de utilizar el prefijo.

Especifique el prefijo de espacio de nombres en un identificador o literal de categoría nacional o alfanumérico.

Se recomienda que el prefijo sea corto, ya que califica el código de inicio y final de cada elemento.

Por ejemplo, considere las siguientes definiciones de datos y sentencia `XML GENERATE` :

```
01  Greeting.
    05  msg  pic x(80)  value 'Hello, world!'.
01  NS  pic x(20)   value 'http://example'.
01  NP  pic x(5)    value 'pre'.
    . . .
    XML Generate Doc from Greeting
        namespace is NS
        namespace-prefix is NP
```

El documento XML resultante tiene un espacio de nombres explícito (`http://example`) y el prefijo `pre` se aplica al código de inicio y final de los elementos `Greeting` y `msg`, como se indica a continuación:

```
<pre:Greeting xmlns:pre="http://example"><pre:msg>Hello, world!</pre:msg></
pre:Greeting>
```

Opcionalmente, puede codificar la frase `NAME` para especificar los nombres de atributo y elemento en el documento XML generado. Los nombres de atributo y elemento deben ser alfanuméricos o literales nacionales y deben ser nombres legales de acuerdo con el estándar XML 1.0.

Por ejemplo, considere la siguiente estructura de datos y sentencia `XML GENERATE` :

```
01 Msg.
    02 Msg-Severity pic 9 value 1.
    02 Msg-Date pic 9999/99/99 value "2012/04/12".
    02 Msg-Text pic X(50) value "Sell everything!".
01 Doc pic X(500).
```

```
    XML Generate Doc from Msg
        With attributes
        Name of Msg          is  "Message"
                Msg-Severity is  "Severity"
                Msg-Date     is  "Date"
                Msg-Text     is  "Text"
    End-XML
```

El documento XML resultante es el siguiente:

```
<Message Severity="1" Date="2012/04/12" Text="Sell everything!"></Message>
```

Opcionalmente, puede codificar la frase SUPPRESS para especificar si se generan elementos de datos individuales basándose en si cumplen o no determinados criterios.

Por ejemplo, considere la siguiente estructura de datos y sentencia XML GENERATE para suprimir espacios y ceros:

```
01 G.
    02 SensitiveInfo.
        03 SSN pic x(11) value '123-45-6789'.
        03 HomeAddress pic x(50) value '123 Main St, Anytown, USA'.
    02 Aarray value spaces.
        03 A pic AAA occurs 5.
    02 Barray value spaces.
        03 B pic XXX occurs 5.
    02 Carray value zeros.
        03 C pic 999 occurs 5.
    Move 'abc' to A(1)
    Move 123 to C(3)
    XML Generate Doc from G
        Suppress SensitiveInfo
                 every nonnumeric element when space
                 every numeric element when zero
    End-XML
```

El documento XML resultante es el siguiente:

```
<G>
   <Aarray><A>abc</A></Aarray>
   <Carray><C>123</C></Carray>
</G>
```

Opcionalmente, puede utilizar la frase TYPE OF para especificar si los elementos de datos individuales se expresan como atributos, elementos o contenido.

Por ejemplo, considere la siguiente estructura de datos y sentencia XML GENERATE :

```
01 Msg.
   02 Msg-Severity pic 9 value 1.
   02 Msg-Date pic 9999/99/99 value "2012/04/12".
   02 Msg-Text pic X(50) value "Sell everything!".
01 Doc pic X(500).
    XML Generate Doc from Msg
        With attributes
        Type of Msg-Severity is attribute
                Msg-Date     is attribute
                Msg-Text     is element
    End-XML
```

El documento XML resultante es el siguiente:

```
<Msg Msg-Severity="1" Msg-Date="2012/04/12">
       <Msg-Text>Sell everything!</Msg-Text></Msg>
```

Además, puede especificar una de las frases siguientes o ambas para recibir el control después de la generación del documento XML:

- ON EXCEPTION, para recibir el control si se produce un error durante la generación de XML
- NOT ON EXCEPTION, para recibir el control si no se produce ningún error

Puede finalizar la sentencia XML GENERATE con el terminador de ámbito explícito END-XML. Codifique END-XML para anidar una sentencia XML GENERATE que tenga la frase ON EXCEPTION o NOT ON EXCEPTION en una sentencia condicional.

La generación de XML continúa hasta que el registro de origen COBOL se ha transformado en XML o se produce un error. Si se produce un error, los resultados son los siguientes:

- El registro especial XML-CODE contiene un código de excepción distinto de cero.
- El control se pasa a la frase ON EXCEPTION , si se especifica, de lo contrario al final de la sentencia XML GENERATE .

Si no se produce ningún error durante la generación de XML, el registro especial XML-CODE contiene cero y el control se pasa a la frase NOT ON EXCEPTION si se especifica o al final de la sentencia XML GENERATE en caso contrario.

"Ejemplo: generación de XML" en la página 446

Conceptos relacionados
Identificador universal de recursos (URI): Sintaxis genérica

Tareas relacionadas
"Control de la codificación de la salida XML generada" en la página 444
"Manejo de excepciones XML GENERATE" en la página 445
"Proceso de datos UTF-8 utilizando tipos de datos UTF-16 (nacional)" en la página 208

Referencias relacionadas
sentencia XML GENERATE (*COBOL for Linux en x86 Consulta de lenguaje*)
XML (Extensible Markup Language)
Espacios de nombres en XML 1.0

Control de la codificación de la salida XML generada

Al generar una salida XML utilizando la sentencia XML GENERATE , puede controlar la codificación de la salida mediante la categoría del elemento de datos que recibe la salida. e identificando la codificación de documentos utilizando la WITH ENCODING frase de la sentencia XML GENERATE .

Acerca de esta tarea

Si especifica la frase WITH ENCODING *codepage* , *codepage* debe identificar una de las páginas de códigos soportadas para el proceso XML de COBOL tal como se describe en la referencia relacionada a continuación sobre la codificación de documentos XML. Si *página de códigos* es un entero, debe ser un número CCSID válido. Si *página de códigos* es de clase alfanumérica o nacional, debe identificar un nombre de página de códigos soportado por las bibliotecas de conversión ICU (International Components for Unicode) tal como se muestra en la tabla de explorador de conversores a la que se hace referencia a continuación.

Si no codifica la frase WITH ENCODING , la salida XML generada se codifica tal como se muestra en la tabla siguiente.

Tabla 43. ***Codificación de XML generado si se omite la frase ENCODING***

Si define el identificador XML de recepción como:	**La salida XML generada se codifica en:**
Alfanumérico nativo (CHAR(EBCDIC) no está en vigor, o la descripción de datos contiene la frase NATIVE)	La página de códigos ASCII o UTF-8 indicada por el entorno local de tiempo de ejecución en vigor
Host alfanumérico (CHAR(EBCDIC) está en vigor y la descripción de datos no contiene la frase NATIVE)	La página de códigos EBCDIC en vigor[1]
Nacional	UTF-16 en formato little-endian

1. Puede establecer la página de códigos EBCDIC utilizando la variable de entorno EBCDIC_CODEPAGE. Si la variable de entorno no está establecida, la codificación está en la página de códigos EBCDIC predeterminada asociada con el entorno local runtime actual.

No se genera una marca de orden de bytes.

Para obtener detalles sobre cómo se convierten los elementos de datos a XML y cómo se forman los nombres de elemento XML y los nombres de atributos a partir de los nombres de datos COBOL, consulte la referencia relacionada a continuación sobre la operación de la sentencia XML GENERATE .

Tareas relacionadas
Capítulo 11, “Establecimiento del entorno local”, en la página 213
“Establecimiento de variables de entorno” en la página 229

Referencias relacionadas
“char” en la página 272
“La codificación de documentos XML” en la página 426
sentencia XML GENERATE (*COBOL for Linux en x86 Consulta de lenguaje*)
Operation of XML GENERATE (*COBOL for Linux en x86 Consulta de lenguaje*)
Componentes internacionales para Unicode: Explorador de conversores

Manejo de excepciones XML GENERATE

Cuando se detecta un error durante la generación de la salida XML, existe una condición de excepción. Puede escribir código para comprobar el registro especial `XML-CODE` , que contiene un código de excepción numérico que indica el tipo de error.

Acerca de esta tarea

Para manejar los errores, utilice una de las frases siguientes de la sentencia `XML GENERATE` o ambas:

- `ON EXCEPTION`
- `COUNT IN`

Si codifica la frase `ON EXCEPTION` en la sentencia `XML GENERATE` , el control se transfiere a la sentencia imperativa que especifique. Puede codificar una sentencia imperativa, por ejemplo, para visualizar el valor `XML-CODE` . Si no codifica una frase `ON EXCEPTION` , el control se transfiere al final de la sentencia `XML GENERATE` .

Cuando se produce un error, un problema puede ser que el elemento de datos que recibe la salida XML no es lo suficientemente grande. En ese caso, la salida XML no está completa y el registro especial `XML-CODE` contiene el código de error 400.

Puede examinar la salida XML generada siguiendo estos pasos:

Procedimiento

1. Codifique la frase `COUNT IN` en la sentencia `XML GENERATE` .

   El campo de recuento que especifique contiene un recuento de las unidades de codificación de caracteres XML que se rellenan durante la generación de XML. Si define la salida XML como nacional, el recuento está en unidades de codificación de caracteres UTF-16 ; para todas las demás codificaciones (incluido para UTF-8), el recuento está en bytes.
2. Utilice el campo de recuento como una longitud de modificación de referencia para hacer referencia a la subserie del elemento de datos de recepción que contiene los caracteres XML que se han generado hasta el punto en el que se ha producido el error.

   Por ejemplo, si `XML-OUTPUT` es el elemento de datos que recibe la salida XML y `XML-CHAR-COUNT` es el campo de recuento, `XML-OUTPUT(1:XML-CHAR-COUNT)` hace referencia a la salida XML.

Resultados

Utilice el contenido de `XML-CODE` para determinar qué acción correctiva se debe llevar a cabo. Para obtener una lista de las excepciones que se pueden producir durante la generación de XML, consulte la referencia relacionada a continuación.

Tareas relacionadas
“Referencia a subseries de elementos de datos” en la página 101

Referencias relacionadas

“excepciones XML GENERATE” en la página 608
XML-CODE (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: generación de XML

El ejemplo siguiente simula la creación de una orden de compra en un elemento de datos de grupo y genera una versión XML de dicha orden de compra.

El programa XGFX utiliza XML GENERATE para producir salida XML en el elemento de datos elemental `xmlPO` del registro de origen, elemento de datos de grupo `purchaseOrder`. Los elementos de datos elementales del registro de origen se convierten al formato de caracteres según sea necesario, y los caracteres se insertan como los valores de los atributos XML cuyos nombres se derivan de los nombres de datos del registro de origen.

XGFX llama al programa `Pretty`, que utiliza la sentencia XML PARSE con procedimiento de proceso p para formatear la salida XML con nuevas líneas y sangría para que el contenido XML se pueda verificar más fácilmente.

Programa XGFX

```
Identification division.
  Program-id. XGFX.
Data division.
 Working-storage section.
   01 numItems pic 99 global.
   01 purchaseOrder global.
     05 orderDate pic x(10).
     05 shipTo.
       10 country pic xx value 'US'.
       10 name pic x(30).
       10 street pic x(30).
       10 city pic x(30).
       10 state pic xx.
       10 zip pic x(10).
     05 billTo.
       10 country pic xx value 'US'.
       10 name pic x(30).
       10 street pic x(30).
       10 city pic x(30).
       10 state pic xx.
       10 zip pic x(10).
     05 orderComment pic x(80).
     05 items occurs 0 to 20 times depending on numItems.
       10 item.
         15 partNum pic x(6).
         15 productName pic x(50).
         15 quantity pic 99.
         15 USPrice pic 999v99.
         15 shipDate pic x(10).
         15 itemComment pic x(40).
   01 numChars comp pic 999.
   01 xmlPO pic x(999).
Procedure division.
  m.
    Move 20 to numItems
    Move spaces to purchaseOrder

    Move '1999-10-20' to orderDate

    Move 'US' to country of shipTo
    Move 'Alice Smith' to name of shipTo
    Move '123 Maple Street' to street of shipTo
    Move 'Mill Valley' to city of shipTo
    Move 'CA' to state of shipTo
    Move '90952' to zip of shipTo

    Move 'US' to country of billTo
    Move 'Robert Smith' to name of billTo
    Move '8 Oak Avenue' to street of billTo
    Move 'Old Town' to city of billTo
    Move 'PA' to state of billTo
    Move '95819' to zip of billTo
    Move 'Hurry, my lawn is going wild!' to orderComment
```

```
    Move 0 to numItems
    Call 'addFirstItem'
    Call 'addSecondItem'
    Move space to xmlPO
    Xml generate xmlPO from purchaseOrder  count in numChars
        with xml-declaration  with attributes
        namespace 'http://www.example.com'  namespace-prefix 'po'
    Call 'pretty' using xmlPO value numChars
    Goback
    .

Identification division.
  Program-id. 'addFirstItem'.
Procedure division.
    Add 1 to numItems
    Move '872-AA' to partNum(numItems)
    Move 'Lawnmower' to productName(numItems)
    Move 1 to quantity(numItems)
    Move 148.95 to USPrice(numItems)
    Move 'Confirm this is electric' to itemComment(numItems)
    Goback.
End program 'addFirstItem'.

Identification division.
  Program-id. 'addSecondItem'.
Procedure division.
    Add 1 to numItems
    Move '926-AA' to partNum(numItems)
    Move 'Baby Monitor' to productName(numItems)
    Move 1 to quantity(numItems)
    Move 39.98 to USPrice(numItems)
    Move '1999-05-21' to shipDate(numItems)
    Goback.
End program 'addSecondItem'.

End program XGFX.
```

Programa bonito

```

Identification division.
  Program-id. Pretty.
Data division.
 Working-storage section.
   01 prettyPrint.
     05 pose pic 999.
     05 posd pic 999.
     05 depth pic 99.
     05 inx pic 999.
     05 elementName pic x(30).
     05 indent pic x(40).
     05 buffer pic x(998).
     05 lastitem pic 9.
       88 unknown value 0.
       88 xml-declaration value 1.
       88 element value 2.
       88 attribute value 3.
       88 charcontent value 4.
 Linkage section.
  1 doc.
   2 pic x occurs 16384 times depending on len.
  1 len comp-5 pic 9(9).
Procedure division using doc value len.
  m.
    Move space to prettyPrint
    Move 0 to depth
    Move 1 to posd pose
    Xml parse doc processing procedure p
    Goback
    .
  p.
    Evaluate xml-event
      When 'VERSION-INFORMATION'
        String '<?xml version="' xml-text '"' delimited by size
            into buffer with pointer posd
        Set xml-declaration to true
      When 'ENCODING-DECLARATION'
        String ' encoding="' xml-text '"' delimited by size
```

```
           into buffer with pointer posd
       When 'STANDALONE-DECLARATION'
         String ' standalone="' xml-text '"' delimited by size
           into buffer with pointer posd
       When 'START-OF-ELEMENT'
         Evaluate true
           When xml-declaration
             String '?>' delimited by size into buffer
                 with pointer posd
             Set unknown to true
             Perform printline
             Move 1 to posd
           When element
             String '>' delimited by size into buffer
                 with pointer posd
           When attribute
             String '">' delimited by size into buffer
                 with pointer posd
         End-evaluate
         If elementName not = space
           Perform printline
         End-if
         Move xml-text to elementName
         Add 1 to depth
         Move 1 to pose
         Set element to true
         String '<' xml-text delimited by size
           into buffer with pointer pose
         Move pose to posd
       When 'ATTRIBUTE-NAME'
         If element
           String ' ' delimited by size into buffer
              with pointer posd
         Else
           String '" ' delimited by size into buffer
              with pointer posd
         End-if
         String xml-text '="' delimited by size into buffer
           with pointer posd
         Set attribute to true
       When 'ATTRIBUTE-CHARACTERS'
         String xml-text delimited by size into buffer
           with pointer posd
       When 'ATTRIBUTE-CHARACTER'
         String xml-text delimited by size into buffer
           with pointer posd
       When 'CONTENT-CHARACTERS'
         Evaluate true
           When element
             String '>' delimited by size into buffer
                 with pointer posd
           When attribute
             String '">' delimited by size into buffer
                 with pointer posd
         End-evaluate
         String xml-text delimited by size into buffer
           with pointer posd
         Set charcontent to true
       When 'CONTENT-CHARACTER'
         Evaluate true
           When element
             String '>' delimited by size into buffer
                 with pointer posd
           When attribute
             String '">' delimited by size into buffer
                 with pointer posd
         End-evaluate
         String xml-text delimited by size into buffer
           with pointer posd
         Set charcontent to true
       When 'END-OF-ELEMENT'
         Move space to elementName
         Evaluate true
           When element
             String '/>' delimited by size into buffer
                 with pointer posd
           When attribute
             String '"/>' delimited by size into buffer
                 with pointer posd
           When other
             String '</' xml-text '>' delimited by size
                 into buffer with pointer posd
```

```
        End-evaluate
        Set unknown to true
        Perform printline
        Subtract 1 from depth
        Move 1 to posd
      When other
        Continue
    End-evaluate
    .
  printline.
    Compute inx = function max(0 2 * depth - 2) + posd - 1
    If inx > 120
      compute inx = 117 - function max(0 2 * depth - 2)
      If depth > 1
        Display indent(1:2 * depth - 2) buffer(1:inx) '...'
      Else
        Display buffer(1:inx) '...'
      End-if
    Else
      If depth > 1
        Display indent(1:2 * depth - 2) buffer(1:posd - 1)
      Else
        Display buffer(1:posd - 1)
      End-if
    End-if
    .
End program Pretty.
```

Salida del programa XGFX

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<po:purchaseOrder xmlns:po="http://www.example.com" orderDate="1999-10-20" orderComment="Hurry, my lawn
is going wild!">
  <po:shipTo country="US" name="Alice Smith" street="123 Maple Street" city="Mill Valley" state="CA"
zip="90952"/>
  <po:billTo country="US" name="Robert Smith" street="8 Oak Avenue" city="Old Town" state="PA"
zip="95819"/>
  <po:items>
    <po:item partNum="872-AA" productName="Lawnmower" quantity="1" USPrice="148.95" shipDate=" "
itemComment="Confirm...
  </po:items>
  <po:items>
    <po:item partNum="926-AA" productName="Baby Monitor" quantity="1" USPrice="39.98"
shipDate="1999-05-21" itemComme...
  </po:items>
</po:purchaseOrder>
```

Tareas relacionadas

Capítulo 19, “Procesando entrada XML”, en la página 419

Referencias relacionadas

Operation of XML GENERATE (*COBOL for Linux en x86 Consulta de lenguaje*)

Mejora de la salida XML

Puede suceder que la información que desea expresar en formato XML ya exista en un elemento de grupo en DATA DIVISION, pero no puede utilizar ese elemento directamente para generar un documento XML debido a uno o varios factores.

Acerca de esta tarea

Por ejemplo:

- Además de los datos necesarios, el elemento tiene elementos de datos subordinados que contienen valores que son irrelevantes para el documento de salida XML.
- Los nombres de los elementos de datos necesarios no son adecuados para la presentación externa, y posiblemente sólo son significativos para los programadores.
- Los elementos de datos necesarios se dividen en demasiados componentes y deben enviarse como salida como contenido del grupo que los contiene.

Hay varias maneras en las que usted puede lidiar con este tipo de situaciones. Una técnica posible es definir un nuevo elemento de datos que tenga las características adecuadas y mover los datos necesarios a los campos adecuados de este nuevo elemento de datos. Sin embargo, este enfoque es algo laborioso y requiere un mantenimiento cuidadoso para mantener sincronizados los elementos de datos originales y los nuevos.

Un enfoque superior que aborda la mayoría de estos problemas es utilizar las nuevas frases opcionales de la sentencia XML GENERATE para:

- Proporcione nombres más significativos y adecuados para los elementos elementales seleccionados y para los elementos de grupo que los contienen.
- Excluya los elementos de datos irrelevantes del XML generado suprimiéndolos en función de sus valores.

El ejemplo al que se hace referencia a continuación muestra una forma de hacerlo.

"Ejemplo: mejora de la salida XML" en la página 450

Referencias relacionadas
Operation of XML GENERATE (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: mejora de la salida XML

El ejemplo siguiente muestra cómo puede modificar la salida XML.

Considere la siguiente estructura de datos. El XML que se genera a partir de la estructura sufre de varios problemas que se pueden corregir.

```
01  CDR-LIFE-BASE-VALUES-BOX.
    15  CDR-LIFE-BASE-VAL-DATE    PIC X(08).
    15  CDR-LIFE-BASE-VALUE-LINE  OCCURS  2 TIMES.
        20  CDR-LIFE-BASE-DESC.
            25 CDR-LIFE-BASE-DESC1 PIC X(15).
            25  FILLER             PIC X(01).
            25  CDR-LIFE-BASE-LIT  PIC X(08).
            25  CDR-LIFE-BASE-DTE  PIC X(08).
        20  CDR-LIFE-BASE-PRICE.
            25  CDR-LIFE-BP-SPACE  PIC 9(08).
            25  CDR-LIFE-BP-DASH   PIC X.
            25  CDR-LIFE-BP-SPACE1 PIC X(02).
        20  CDR-LIFE-BASE-PRICE-ED  REDEFINES
             CDR-LIFE-BASE-PRICE  PIC $$$.$$.
        20  CDR-LIFE-BASE-QTY.
            25  CDR-LIFE-QTY-SPACE   PIC X(08).
            25  CDR-LIFE-QTY-DASH    PIC X.
            25  CDR-LIFE-QTY-SPACE1  PIC X(03).
            25  FILLER               PIC X(02).
        20  CDR-LIFE-BASE-VALUE   PIC $$$9.99
            BLANK WHEN ZERO.
    15  CDR-LIFE-BASE-TOT-VALUE   PIC X(15)
```

Cuando esta estructura de datos se llena con algunos valores de ejemplo, y XML se genera directamente a partir de ella y, a continuación, se formatea utilizando el programa `Pretty` (que se muestra en la "Ejemplo: generación de XML" en la página 446), el resultado es el siguiente:

```
<CDR-LIFE-BASE-VALUES-BOX>
  <CDR-LIFE-BASE-VAL-DATE>01/02/03</CDR-LIFE-BASE-VAL-DATE>
  <CDR-LIFE-BASE-VALUE-LINE>
    <CDR-LIFE-BASE-DESC>
      <CDR-LIFE-BASE-DESC1>First</CDR-LIFE-BASE-DESC1>
      <CDR-LIFE-BASE-LIT> </CDR-LIFE-BASE-LIT>
      <CDR-LIFE-BASE-DTE>01/01/01</CDR-LIFE-BASE-DTE>
    </CDR-LIFE-BASE-DESC>
    <CDR-LIFE-BASE-PRICE>
      <CDR-LIFE-BP-SPACE>23</CDR-LIFE-BP-SPACE>
      <CDR-LIFE-BP-DASH>.</CDR-LIFE-BP-DASH>
      <CDR-LIFE-BP-SPACE1>00</CDR-LIFE-BP-SPACE1>
    </CDR-LIFE-BASE-PRICE>
    <CDR-LIFE-BASE-QTY>
      <CDR-LIFE-QTY-SPACE>123</CDR-LIFE-QTY-SPACE>
```

```
      <CDR-LIFE-QTY-DASH>.</CDR-LIFE-QTY-DASH>
      <CDR-LIFE-QTY-SPACE1>000</CDR-LIFE-QTY-SPACE1>
    </CDR-LIFE-BASE-QTY>
    <CDR-LIFE-BASE-VALUE>$765.00</CDR-LIFE-BASE-VALUE>
  </CDR-LIFE-BASE-VALUE-LINE>
  <CDR-LIFE-BASE-VALUE-LINE>
    <CDR-LIFE-BASE-DESC>
      <CDR-LIFE-BASE-DESC1>Second</CDR-LIFE-BASE-DESC1>
      <CDR-LIFE-BASE-LIT> </CDR-LIFE-BASE-LIT>
      <CDR-LIFE-BASE-DTE>02/02/02</CDR-LIFE-BASE-DTE>
    </CDR-LIFE-BASE-DESC>
    <CDR-LIFE-BASE-PRICE>
      <CDR-LIFE-BP-SPACE>34</CDR-LIFE-BP-SPACE>
      <CDR-LIFE-BP-DASH>.</CDR-LIFE-BP-DASH>
      <CDR-LIFE-BP-SPACE1>00</CDR-LIFE-BP-SPACE1>
    </CDR-LIFE-BASE-PRICE>
    <CDR-LIFE-BASE-QTY>
      <CDR-LIFE-QTY-SPACE>234</CDR-LIFE-QTY-SPACE>
      <CDR-LIFE-QTY-DASH>.</CDR-LIFE-QTY-DASH>
      <CDR-LIFE-QTY-SPACE1>000</CDR-LIFE-QTY-SPACE1>
    </CDR-LIFE-BASE-QTY>
    <CDR-LIFE-BASE-VALUE>$654.00</CDR-LIFE-BASE-VALUE>
  </CDR-LIFE-BASE-VALUE-LINE>
  <CDR-LIFE-BASE-TOT-VALUE>Very high!</CDR-LIFE-BASE-TOT-VALUE>
</CDR-LIFE-BASE-VALUES-BOX>
```

Este XML generado tiene varios problemas:

- Los nombres de elemento son largos y no muy significativos.
- Algunos campos que son elementos deben ser atributos como, por ejemplo, `CDR-LIFE-BASE-VAL-DATE` y `CDR-LIFE-BASE-DESC1`.
- Hay datos no deseados, por ejemplo, `CDR-LIFE-BASE-LIT` y `CDR-LIFE-BASE-DTE`.
- Los datos necesarios tienen un padre innecesario. Por ejemplo, `CDR-LIFE-BASE-DESC1` tiene padre `CDR-LIFE-BASE-DESC`.
- Otros campos obligatorios se dividen en demasiados subcomponentes. Por ejemplo, `CDR-LIFE-BASE-PRICE` tiene tres subcomponentes para una cantidad.

Estas y otras características de la salida XML se pueden corregir utilizando frases adicionales de la sentencia `XML GENERATE` como se indica a continuación:

- Utilice la frase `NAME OF` para proporcionar los nombres de etiqueta o atributo adecuados.
- Utilice la frase `TYPE OF … IS ATTRIBUTE` para seleccionar los campos que deben ser atributos XML en lugar de elementos.
- Utilice la frase `TYPE OF … IS CONTENT` para suprimir etiquetas para subcomponentes excesivos.
- Utilice la frase `SUPPRESS … WHEN` para excluir los campos que contienen valores poco interesantes.

A continuación se muestra un ejemplo de la sentencia `XML GENERATE` para solucionar estos problemas:

```
XML generate Doc from CDR-LIFE-BASE-VALUES-BOX
  Count in tally
  Name of
         CDR-LIFE-BASE-VALUES-BOX
       is 'Base_Values'
         CDR-LIFE-BASE-VAL-DATE
       is 'Date'
         CDR-LIFE-BASE-DTE
       is 'Date'
         CDR-LIFE-BASE-VALUE-LINE
       is 'BaseValueLine'
         CDR-LIFE-BASE-DESC1
       is 'Description'
         CDR-LIFE-BASE-PRICE
       is 'BasePrice'
         CDR-LIFE-BASE-QTY
       is 'BaseQuantity'
         CDR-LIFE-BASE-VALUE
       is 'BaseValue'
         CDR-LIFE-BASE-TOT-VALUE
       is 'TotalValue'
  Type of
         CDR-LIFE-BASE-VAL-DATE is attribute
```

```
          CDR-LIFE-BASE-DESC1 is attribute
          CDR-LIFE-BP-SPACE   is content
          CDR-LIFE-BP-DASH    is content
          CDR-LIFE-BP-SPACE1  is content
          CDR-LIFE-QTY-SPACE  is content
          CDR-LIFE-QTY-DASH   is content
          CDR-LIFE-QTY-SPACE1 is content
  Suppress every nonnumeric when space
           every numeric when zero
```

El resultado de generar y formatear XML a partir de la sentencia mostrada anteriormente es más utilizable:

```
<Base_Values Date="01/02/03">
  <BaseValueLine Description="First">
    <Date>01/01/01</Date>
    <BasePrice>23.00</BasePrice>
    <BaseQuantity>123.000</BaseQuantity>
    <BaseValue>$765.00</BaseValue>
  </BaseValueLine>
  <BaseValueLine Description="Second">
    <Date>02/02/02</Date>
    <BasePrice>34.00</BasePrice>
    <BaseQuantity>234.000</BaseQuantity>
    <BaseValue>$654.00</BaseValue>
  </BaseValueLine>
  <TotalValue>Very high!</TotalValue>
</Base_Values>
```

Tenga en cuenta que la palabra reservada COBOL DATE ahora se puede utilizar como un nombre de código XML en la salida y que también se pueden utilizar otros caracteres que no están permitidos en los nombres de datos COBOL, como por ejemplo y el subrayado _ .

Tenga en cuenta que la palabra reservada de COBOL DATE ahora se puede utilizar como un nombre de código XML en la salida. También se pueden utilizar caracteres como las letras acentuadas y el punto . que no están permitidos en los nombres de datos de un solo byte.

Referencias relacionadas
Operation of XML GENERATE (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia REPLACE (*COBOL for Linux en x86 Consulta de lenguaje*)

Parte 6. Cómo trabajar con aplicaciones más complejas

Capítulo 21. Portabilidad de aplicaciones entre plataformas y compiladores COBOL

El sistema Linux en x86 tiene un hardware y una arquitectura de sistema operativo diferentes a los de un sistema IBM Z o IBM Potencia . Debido a estas diferencias, pueden surgir algunos problemas al mover programas COBOL entre estos entornos.

Acerca de esta tarea

Las tareas relacionadas y la referencia a continuación describen algunas de las diferencias entre plataformas de desarrollo y proporcionan instrucciones para ayudarle a minimizar los problemas de portabilidad.

Tareas relacionadas
"Migración de una versión anterior de COBOL for Linux on x86 a la versión actual" en la *Guía de migración*
"Migración de compiladores noIBM COBOL a COBOL for Linux on x86" en la *Guía de migración*
"Migración desde Enterprise COBOL for z/OS a COBOL for Linux on x86" en la *Guía de migración*
"Migración de COBOL for AIX a COBOL for Linux on x86" en la *Guía de migración*

Capítulo 22. Utilización de subprogramas

Acerca de esta tarea

Muchas aplicaciones constan de varios programas compilados por separado que están enlazados entre sí. Si los programas se llaman unos a otros, deben poder comunicarse. Necesitan transferir el control y normalmente necesitan acceso a datos comunes.

Los programas COBOL anidados entre sí también se pueden comunicar. Todos los subprogramas necesarios para una aplicación pueden estar en un archivo fuente y, por lo tanto, sólo requieren una compilación.

Conceptos relacionados
“Programas principales, subprogramas y llamadas” en la página 457

Tareas relacionadas
“Finalización y reintroducción de programas o subprogramas principales” en la página 457
“Llamada a programas COBOL anidados” en la página 458
“Llamada a programas COBOL no anidados” en la página 462
“Llamada entre programas COBOL y C/C++” en la página 464
“Realización de llamadas recursivas” en la página 470

Programas principales, subprogramas y llamadas

Si un programa COBOL es el primer programa de una unidad de ejecución, ese programa COBOL es el *programa principal*. De lo contrario, él y todos los demás programas COBOL de la unidad de ejecución son *subprogramas*. Ninguna sentencia u opción de código fuente específica identifica un programa COBOL como programa o subprograma principal.

Si un programa COBOL es un programa o subprograma principal puede ser significativo por cualquiera de dos razones:

- Efecto de las sentencias de terminación de programa
- Estado del programa cuando se vuelve a especificar después de devolver

En `PROCEDURE DIVISION`, un programa puede llamar a otro programa (generalmente denominado *subprograma*), y este programa llamado puede llamar a otros programas. El programa que llama a otro programa se conoce como el programa *que llama* , y el programa al que llama se conoce como el programa *llamado* . Cuando se completa el proceso del programa llamado, el programa llamado puede transferir el control de vuelta al programa de llamada o finalizar la unidad de ejecución.

El programa COBOL llamado empieza a ejecutarse en la parte superior de `PROCEDURE DIVISION`.

Tareas relacionadas
“Finalización y reintroducción de programas o subprogramas principales” en la página 457
“Llamada a programas COBOL anidados” en la página 458
“Llamada a programas COBOL no anidados” en la página 462
“Llamada entre programas COBOL y C/C++” en la página 464
“Realización de llamadas recursivas” en la página 470

Finalización y reintroducción de programas o subprogramas principales

Si un programa se deja en su último estado utilizado o en su estado inicial, y a qué llamante devuelve, puede depender de las sentencias de terminación que utilice.

Acerca de esta tarea

Para finalizar la ejecución en el programa principal, debe codificar una sentencia STOP RUN o GOBACK en el programa principal. STOP RUN termina la unidad de ejecución y cierra todos los archivos abiertos por el programa principal y sus subprogramas llamados. El control se devuelve al interlocutor del programa principal, que a menudo es el sistema operativo. GOBACK tiene el mismo efecto en el programa principal. Un EXIT PROGRAM realizado en un programa principal no tiene ningún efecto.

Puede finalizar un subprograma utilizando una sentencia EXIT PROGRAM, GOBACKo STOP RUN . Si utiliza una sentencia EXIT PROGRAM o GOBACK , el control vuelve al interlocutor inmediato del subprograma sin que finalice la unidad de ejecución. Se genera una sentencia EXIT PROGRAM implícita si no hay una siguiente sentencia ejecutable en un programa llamado. Si finaliza el subprograma con una sentencia STOP RUN , el efecto es el mismo que en un programa principal: todos los programas COBOL de la unidad de ejecución terminan y el control vuelve al llamante del programa principal.

Normalmente, un subprograma se deja en su *último estado utilizado* cuando termina con EXIT PROGRAM o GOBACK. La próxima vez que se llame al subprograma en la unidad de ejecución, sus valores internos serán los que se dejaron, excepto que los valores de retorno para las sentencias PERFORM se restablecerán a sus valores iniciales. (En cambio, un programa principal se inicializa cada vez que se llama.)

Hay algunos casos en los que los programas estarán en su estado inicial:

- Un subprograma que se llama dinámicamente y luego se cancela estará en el estado inicial la próxima vez que se llame.
- Un programa que tenga la cláusula INITIAL en el párrafo PROGRAM-ID estará en el estado inicial cada vez que se llame.
- Los elementos de datos definidos en LOCAL-STORAGE SECTION se restablecerán al estado inicial especificado por sus cláusulas VALUE cada vez que se llame al programa.

Conceptos relacionados
"Comparación de WORKING-STORAGE y LOCAL-STORAGE" en la página 11

Tareas relacionadas
"Llamada a programas COBOL anidados" en la página 458
"Realización de llamadas recursivas" en la página 470

Llamada a programas COBOL anidados

Al llamar a programas anidados, puede crear aplicaciones que utilicen técnicas de programación estructurada. También puede llamar a programas anidados en lugar de procedimientos PERFORM para evitar la modificación no intencionada de elementos de datos.

Acerca de esta tarea

Utilice sentencias CALL *literal* o CALL *identificador* para realizar llamadas a programas anidados.

Puede llamar a un programa anidado sólo desde su programa que lo contiene directamente, a menos que identifique el programa anidado como COMMON en su párrafo PROGRAM-ID . En ese caso, puede llamar al *programa común* desde cualquier programa que sea anidado (directa o indirectamente) en el mismo programa que el programa común. Solo los programas anidado se pueden identificar como COMMON. Las llamadas recursivas no están permitidas.

Siga estas directrices cuando utilice estructuras de programa anidadas:

- Codifique un IDENTIFICATION DIVISION en cada programa. Todas las demás divisiones son opcionales.
- Opcionalmente, haga que el nombre de cada programa de anidado sea exclusivo. Aunque no es necesario que los nombres de los programas anidado sean exclusivos (tal como se describe en la referencia relacionada sobre el ámbito de nombres), hacer que los nombres sean exclusivos podría

ayudar a que la aplicación sea más mantenible. Puede utilizar cualquier palabra definida por el usuario válida o un literal alfanumérico como nombre de un programa anidado .

- En el programa más externo, codifique las entradas `CONFIGURATION SECTION` que puedan ser necesarias. Los programas Anidado no pueden tener un `CONFIGURATION SECTION`.
- Incluya cada programa anidado en el programa contenedor inmediatamente antes del marcador `END PROGRAM` del programa contenedor.
- Utilice un marcador `END PROGRAM` para terminar anidado y los programas que lo contienen.

Conceptos relacionados
"Programas anidados" en la página 459

Referencias relacionadas
"Ámbito de nombres" en la página 461

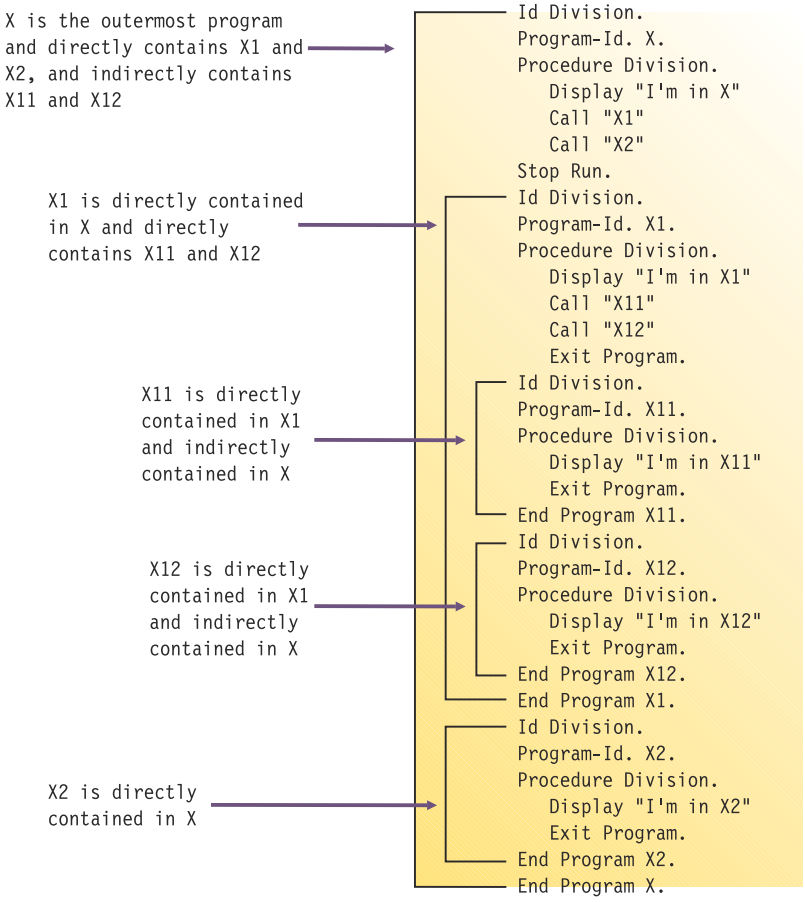
Programas anidados

Un programa COBOL puede *anidar*o contener otros programas COBOL. Los propios programas anidados pueden contener otros programas. Un programa anidado puede estar contenido directa o indirectamente en un programa.

Existen cuatro ventajas principales para anidar los programas llamados:

- Los programas anidados proporcionan un método para crear funciones modulares y mantener técnicas de programación estructurada. Se pueden utilizar de forma análoga para realizar procedimientos (utilizando la sentencia `PERFORM`), pero con un flujo de control más estructurado y con la capacidad de proteger los elementos de datos locales.
- Los programas anidados le permiten depurar un programa antes de incluirlo en una aplicación.
- Los programas anidados le permiten compilar una aplicación con una sola invocación del compilador.
- Las llamadas a programas anidados tienen el mejor rendimiento de todas las formas de sentencias COBOL `CALL` .

El ejemplo siguiente describe una estructura anidada que contiene directa e indirectamente programas:



"Ejemplo: estructura de programas anidados" en la página 460

Tareas relacionadas

"Llamada a programas COBOL anidados" en la página 458

Referencias relacionadas

"Ámbito de nombres" en la página 461

Ejemplo: estructura de programas anidados

El ejemplo siguiente muestra una estructura anidada con algunos programas anidado identificados como COMMON.

```
Program-Id. A.
  Program-Id. A1.
    Program-Id. A11.
      Program-Id. A111.
      End Program A111.
    End Program A11.
    Program-Id. A12 is Common.
    End Program A12.
  End Program A1.
  Program-Id. A2 is Common.
  End Program A2.
  Program-Id. A3 is Common.
  End Program A3.
End Program A.
```

La tabla siguiente describe la jerarquía de llamada para la estructura que se muestra en el ejemplo anterior. Los programas A12, A2y A3 se identifican como COMMON, y las llamadas asociadas con ellos difieren.

Este programa	Puede llamar a estos programas	Y pueden ser llamados por estos programas
A	A1, A2, A3	Ninguna
A1	A11, A12, A2, A3	A
A11	A111, A12, A2, A3	A1
A111	A12, A2, A3	A11
A12	A2, A3	A1, A11, A111
A2	A3	A, A1, A11, A111, A12, A3
A3	A2	A, A1, A11, A111, A12, A2

En este ejemplo, tenga en cuenta que:

- A2 no puede llamar a A1 porque A1 no es común y no está contenido en A2.
- A1 puede llamar a A2 porque A2 es común.

Ámbito de nombres

Los nombres de las estructuras anidadas se dividen en dos clases: local y global. La clase determina si un nombre se conoce más allá del ámbito del programa que lo declara. Una secuencia de búsqueda específica localiza la declaración de un nombre después de que se haga referencia a él en un programa.

Nombres locales

Los nombres (excepto el nombre de programa) son locales a menos que se declare lo contrario. Los nombres locales sólo son visibles o accesibles dentro del programa en el que se declaran. No son visibles ni accesibles para los programas contenidos y que los contienen.

Nombres globales

Un nombre que es global (indicado utilizando la cláusula GLOBAL) es visible y accesible para el programa en el que se declara y para todos los programas que están directa e indirectamente contenidos en dicho programa. Por lo tanto, los programas contenidos pueden compartir datos y archivos comunes del programa que los contiene simplemente haciendo referencia a los nombres de los elementos.

Cualquier elemento que esté subordinado a un elemento global (incluidos los nombres de condición e índices) es automáticamente global.

Puede declarar el mismo nombre con la cláusula GLOBAL más de una vez, siempre que cada declaración se produzca en un programa diferente. Tenga en cuenta que puede enmascarar u ocultar un nombre en una estructura anidada haciendo que el mismo nombre aparezca en distintos programas de la misma estructura contenedora. Sin embargo, este enmascaramiento podría causar problemas durante una búsqueda de una declaración de nombre.

Busca declaraciones de nombre

Cuando se hace referencia a un nombre en un programa, se realiza una búsqueda para localizar la declaración para ese nombre. La búsqueda empieza en el programa que contiene la referencia y continúa hacia fuera a los programas que la contienen hasta que se encuentra una coincidencia. La búsqueda sigue este proceso:

1. Se busca en las declaraciones del programa.
2. Si no se encuentra ninguna coincidencia, sólo se busca en las declaraciones globales en los sucesivos programas que las contienen.
3. La búsqueda finaliza cuando se encuentra el primer nombre coincidente. Si no se encuentra ninguna coincidencia, existe un error.

La búsqueda es para un nombre global, no para un tipo determinado de objeto asociado con el nombre como, por ejemplo, un elemento de datos o un conector de archivo. La búsqueda se detiene cuando se encuentra alguna coincidencia, independientemente del tipo de objeto. Si el objeto declarado es de un tipo distinto al esperado, existe una condición de error.

Llamada a programas COBOL no anidados

Un programa COBOL puede llamar a un subprograma que está enlazado en el mismo módulo ejecutable que el llamante (*enlace estático*) o que se proporciona en un biblioteca compartida (*enlace dinámico*). COBOL para Linux también proporciona la resolución en tiempo de ejecución de un subprograma de destino desde un biblioteca compartida.

Acerca de esta tarea

Si enlaza un programa de destino estáticamente, forma parte del módulo ejecutable del llamante y se carga con el llamante. Si enlaza dinámicamente o resuelve una llamada en tiempo de ejecución, el programa de destino se proporciona en una biblioteca y se carga cuando se carga el llamante o cuando se llama al programa de destino.

El enlace dinámico o estático de subprogramas se realiza para COBOL `CALL` *literal*. La resolución de tiempo de ejecución siempre se realiza para COBOL `CALL` *identificador* y se realiza para `CALL` *literal* si la opción `DYNAM` está en vigor.

Restricción: No puede mezclar programas COBOL de 32 bits y 64 bits en una aplicación. Todos los componentes de programa de una aplicación deben compilarse utilizando el mismo valor de la opción de compilador `ADDR` .

Conceptos relacionados
“Identificador CALL y literal CALL” en la página 462
“Enlace estático frente a utilización de bibliotecas compartidas” en la página 489

Referencias relacionadas
“ADDR” en la página 268
“DYNAM” en la página 281
sentencia CALL (*COBOL for Linux en x86 Consulta de lenguaje*)

Identificador CALL y literal CALL

`CALL` *identificador*, donde *identificador* es un elemento de datos que contiene el nombre de un subprograma no anidado en tiempo de ejecución, siempre hace que el subprograma de destino se cargue cuando se llama. `CALL` *literal*, donde *literal* es el nombre explícito de un subprograma de destino no anidado, se puede resolver de forma estática o dinámica.

Con `CALL` *identificador*, el nombre del ejecutable o biblioteca compartida debe coincidir con el nombre del punto de entrada de destino.

Con `CALL` *literal*, si la opción de compilador `NODYNAM` está en vigor, se puede realizar el enlace estático o dinámico. Si `DYNAM` está en vigor, `CALL` *literal* se resuelve de la misma forma que `CALL` *identificador*: el subprograma de destino se carga cuando se llama, y el nombre del ejecutable debe coincidir con el nombre del punto de entrada de destino.

Estas definiciones de llamada sólo se aplican en el caso de un programa COBOL que llama a un programa no anidado. Si un programa COBOL llama a un programa anidado, el compilador resuelve la llamada sin ninguna intervención del sistema.

Limitación: Dos o más ejecutables enlazados por separado en una aplicación no debe llamar estáticamente a el mismo subprograma no anidado.

Conceptos relacionados
“Enlace estático frente a utilización de bibliotecas compartidas” en la página 489

Referencias relacionadas
“DYNAM” en la página 281
sentencia CALL (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: llamada dinámica utilizando el identificador CALL

El ejemplo siguiente muestra cómo puede realizar llamadas dinámicas que utilizan el identificador CALL .

El primer programa, dl1.cbl, utiliza el identificador de CALL para llamar al segundo programa, dl1a.cbl.

dl1.cbl

```
* Simple dynamic call to dl1a

 Identification Division.
 Program-id.    dl1.
*
 Environment Division.
 Configuration Section.
 Input-Output Section.
 File-control.
*
 Data Division.
 File Section.
 Working-storage Section.
 01 var pic x(10).
 Linkage Section.
*
 Procedure Division.
     move "Dl1A" to var.
     display "Calling " var.
     call var.
     move "dl1a     " to var.
     display "Calling " var.
     call var.
     stop run.
 End program dl1.
```

dlla.cbl

```
* Called by dl1.cbl using CALL identifier.

 IDENTIFICATION DIVISION.
 PROGRAM-ID. dl1a.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 OBJECT-COMPUTER. ANY-THING.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77 num pic 9(4) binary value is zero.
*
 PROCEDURE DIVISION.
 LA-START.
     display "COBOL DL1A function." upon console.
     add 11 to num.
     display "num = " num
     goback.
```

Procedimiento

Para crear y ejecutar el ejemplo anterior, siga estos pasos:

1. Especifique cob2 dl1.cbl -o dl1 para generar el módulo ejecutable dl1.
2. Especifique cob2 dl1a.cbl -dll -o DL1A para generar el módulo ejecutable DL1A.

   A menos que compile utilizando la opción PGMNAME(MIXED) , los nombres de programa ejecutables se cambian a mayúsculas.

`-dll`, `-dso` y `-shared` son tres opciones equivalentes, y aquí se puede especificar cualquiera. Consulte -dll | -dso | -shared para obtener más información sobre estas opciones.

3. Especifique el mandato `export COBPATH=.` para que se busque en el directorio actual los destinos de las llamadas dinámicas.
4. Especifique `dl1` para ejecutar el programa.

Puesto que el destino `CALL` *identificador* debe coincidir con el nombre del programa llamado, el módulo ejecutable del ejemplo anterior se genera como `DL1A`, no como `dl1a`.

Referencias relacionadas
“PGMNAME” en la página 292

Llamada entre programas COBOL y C/C++

Puede llamar a funciones escritas en C/C++ desde programas COBOL y puede llamar a programas COBOL desde funciones de C/C++ .

Acerca de esta tarea

En una aplicación entre lenguajes, puede combinar programas COBOL de 64 bits con funciones de C/C++ de 64 bits o programas COBOL de 32 bits con funciones de C/C++ de 32 bits.

Restricción: No puede mezclar componentes de 32 bits y componentes de 64 bits en una aplicación.

Comunicación entre lenguajes entre COBOL y C++: En una aplicación entre lenguajes que mezcla COBOL y C++, siga estas directrices:

- Especifique `extern "C"` en prototipos de función para programas COBOL a los que se llama desde C++, y en funciones C++ a las que se llama desde COBOL.
- En COBOL, utilice los parámetros `BY VALUE` para que coincidan con el convenio de parámetros C++ normal.
- En C++, utilice parámetros de referencia para que coincidan con el convenio COBOL `BY REFERENCE` .
- Los programas COBOL siguen el enlace C y C++ y los convenios de llamada. Los programas C y C++ que llamarán dinámicamente los programas COBOL se deben crear como DSOs (objetos compartidos dinámicos) compilándolos utilizando la opción **`-fPIC`** de gcc para emitir código independiente de la posición que sea adecuado para el enlace dinámico y enlazándolos con la opción **`-shared`** de gcc para producir un objeto compartido que luego se pueda enlazar con otros objetos para formar un ejecutable.

Las reglas y directrices a las que se hace referencia a continuación proporcionan información adicional sobre cómo realizar estas llamadas entre idiomas.

Referencias no calificadas a "C/C++" en las secciones referenciadas son para compilador GNU GCC.

Tareas relacionadas
“Llamada entre COBOL y C/C++ en CICS” en la página 413
“Inicialización de entornos” en la página 464
“Pasar datos entre COBOL y C/C++” en la página 465
“Contraer marcos de pila y terminar unidades de ejecución o procesos” en la página 465
Capítulo 23, “compartir datos”, en la página 473

Referencias relacionadas
“ADDR” en la página 268
“Tipos de datos COBOL y C/C++” en la página 466

Inicialización de entornos

Para realizar una llamada entre C/C++ y COBOL, debe inicializar correctamente el entorno de destino.

Acerca de esta tarea

Si el programa principal está escrito en C/C++ y realiza varias llamadas a un programa COBOL, utilice uno de los métodos siguientes:

- Preinicialice el entorno COBOL en el programa C/C++ antes de llamar a cualquier programa COBOL. Este enfoque se recomienda porque proporciona el mejor rendimiento.
- Coloque el programa COBOL en un ejecutable que no forme parte de la rutina de C/C++ que llama a COBOL. A continuación, cada vez que desee llamar al programa COBOL principal, realice los pasos siguientes en el programa C/C++ :
  1. Cargue el programa.
  2. Llame al programa.
  3. Descargue el programa.

Conceptos relacionados
Capítulo 25, “Preinicialización del entorno de ejecución COBOL”, en la página 495

Pasar datos entre COBOL y C/C++

Algunos tipos de datos COBOL tienen C/C++ equivalentes, pero otros no. Cuando pase datos entre programas COBOL y funciones de C/C++ , asegúrese de limitar el intercambio de datos a los tipos de datos adecuados.

Acerca de esta tarea

De forma predeterminada, COBOL pasa los argumentos `BY REFERENCE`. Si pasa un argumento `BY REFERENCE`, C/C++ obtiene un puntero al argumento. Si pasa un argumento `BY VALUE`, COBOL pasa el argumento real. Solo puede utilizar `BY VALUE` para los siguientes tipos de datos:

- Un carácter alfanumérico
- Un carácter `USAGE NATIONAL`
- `BINARY`
- `COMP`
- `COMP-1`
- `COMP-2`
- `COMP-4`
- `COMP-5`
- `FUNCTION-POINTER`
- `POINTER`
- `PROCEDURE-POINTER`

“Ejemplo: Programa COBOL que llama a funciones C” en la página 467
“Ejemplo: programas C a los que llama y llama COBOL” en la página 468
“Ejemplo: Programa COBOL que llama a la función C++” en la página 469

Tareas relacionadas
Capítulo 23, “compartir datos”, en la página 473

Referencias relacionadas
“Tipos de datos COBOL y C/C++” en la página 466

Contraer marcos de pila y terminar unidades de ejecución o procesos

No invoque funciones en un lenguaje que contraiga marcos de pila de programa de otro lenguaje.

Acerca de esta tarea

Esta directriz incluye estas situaciones:

- Contraer algunos marcos de pila activos de un idioma cuando hay marcos de pila activos escritos en otro idioma en los marcos de pila que se van a contraer (C/C++ longjmp ()).
- La terminación de una unidad de ejecución o un proceso desde un lenguaje mientras que los marcos de pila escritos en otro lenguaje están activos, como por ejemplo la emisión de un COBOL STOP RUN o un C/C++ exit () o _exit (). En su lugar, estructure la aplicación de tal forma que un programa invocado termine volviendo a su invocador.

Puede utilizar las llamadas C/C++ longjmp () o COBOL STOP RUN y C/C++ exit () o _exit () si al hacerlo no contrae marcos de pila activos de un lenguaje que no sea el lenguaje que inicia esa acción. Para los idiomas que no inician la contracción y la terminación, de lo contrario podrían producirse estos efectos adversos:

- Es posible que no se realicen las funciones de limpieza o salida normales del lenguaje, como por ejemplo el cierre de archivos por COBOL durante la terminación de la unidad de ejecución, o la limpieza de recursos adquiridos dinámicamente por el lenguaje terminado involuntariamente.
- Es posible que no se invoquen las salidas o funciones especificadas por el usuario para la salida o la terminación, como por ejemplo destructores y la función atexit () de C/C++ .

En general, las excepciones incurridas durante la ejecución de un marco de pila se manejan de acuerdo con las reglas del lenguaje que incurre en la excepción. Puesto que la implementación de COBOL no depende de la interceptación de excepciones a través de servicios del sistema para el soporte de la semántica de lenguaje COBOL, puede especificar la opción de tiempo de ejecución TRAP(OFF) para habilitar la semántica de manejo de excepciones del lenguaje no COBOL.

COBOL para Linux guarda el entorno de excepción en la inicialización del entorno de ejecución COBOL y lo restaura en la terminación del entorno COBOL. COBOL espera que los lenguajes de interfaz y las herramientas sigan el mismo convenio.

Referencias relacionadas
"TRAP" en la página 318

Tipos de datos COBOL y C/C++

La tabla siguiente muestra la correspondencia entre los tipos de datos que están disponibles en COBOL y C/C++.

Tabla 44. ***Tipos de datos COBOL y C/C++***

Tipos de datos de C/C++	**Tipos de datos COBOL**
Wchar_t	USAGE NATIONAL (PICTURE N)
char	PIC X
carácter firmado	No hay ningún equivalente COBOL adecuado
unsigned char	No hay ningún equivalente COBOL adecuado
int firmado corto	PIC S9-S9(4) COMP-5. Puede ser COMP, COMP-4o BINARY si utiliza la opción de compilador TRUNC(BIN) .
int corto sin firmar	PIC 9-9(4) COMP-5. Puede ser COMP, COMP-4o BINARY si utiliza la opción de compilador TRUNC(BIN) .
int largo	PIC 9(5)-9(9) COMP-5. Puede ser COMP, COMP-4o BINARY si utiliza la opción de compilador TRUNC(BIN) .

*Tabla 44. **Tipos de datos COBOL y C/C++** (continuación)*	
Tipos de datos de C/C++	**Tipos de datos COBOL**
long long int	`PIC 9(10)-9(18) COMP-5`. Puede ser `COMP`, `COMP-4`o `BINARY` si utiliza la opción de compilador `TRUNC(BIN)` .
float	`COMP-1`
doble	`COMP-2`
enumeración	Análogo al nivel 88, pero no idéntico
char (*n*)	`PICTURE X(n)`
puntero de matriz (*) a tipo	No hay ningún equivalente COBOL adecuado
puntero (*) a función	`PROCEDURE-POINTER` o `FUNCTION-POINTER`

Tareas relacionadas
"Pasar datos entre COBOL y C/C++" en la página 465

Referencias relacionadas
"TRUNC" en la página 302

Ejemplo: Programa COBOL que llama a funciones C

El ejemplo siguiente muestra un programa COBOL que llama a funciones C utilizando la sentencia `CALL` .

El ejemplo ilustra los conceptos siguientes:

- La sentencia `CALL` no indica si el programa llamado está escrito en COBOL o C.
- COBOL da soporte a la llamada a programas que tienen nombres con mayúsculas y minúsculas.
- Puede pasar argumentos a programas C de varias formas (por ejemplo, `BY REFERENCE` o `BY VALUE`).
- Debe declarar un valor de retorno de función en una sentencia `CALL` que llame a una función non-void C .
- Debe correlacionar los tipos de datos COBOL con los tipos de datos C adecuados.

```
 CBL PGMNAME(MIXED)
* This compiler option allows for case-sensitive names for called programs.
*
 IDENTIFICATION DIVISION.
 PROGRAM-ID. "COBCALLC".
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 N4              PIC 9(4)  COMP-5.
 01 NS4             PIC S9(4) COMP-5.
 01 N9              PIC 9(9)  COMP-5.
 01 NS9             PIC S9(9) COMP-5.
 01 NS18            USAGE COMP-2.
 01 D1              USAGE COMP-2.
 01 D2              USAGE COMP-2.
 01 R1.
     02 NR1         PIC 9(8) COMP-5.
     02 NR2         PIC 9(8) COMP-5.
     02 NR3         PIC 9(8) COMP-5.
 PROCEDURE DIVISION.
     MOVE 123 TO N4
     MOVE -567 TO NS4
     MOVE 98765432 TO N9
     MOVE -13579456 TO NS9
     MOVE 222.22 TO NS18
     DISPLAY "Call MyFun with n4=" N4 " ns4=" NS4 " N9=" n9
     DISPLAY "                 ns9="  NS9" ns18="  NS18
* The following CALL illustrates several ways to pass arguments.
*
     CALL "MyFun" USING N4 BY VALUE NS4 BY REFERENCE N9 NS9 NS18
     MOVE 1024 TO N4
```

```
* The following CALL returns the C function return value.
*
     CALL "MyFunR" USING BY VALUE N4 RETURNING NS9
     DISPLAY "n4=" N4 " and ns9= n4 times n4= " NS9
     MOVE -357925680.25 TO D1
     CALL "MyFunD" USING BY VALUE D1 RETURNING D2
     DISPLAY "d1=" D1 " and d2= 2.0 times d2= " D2
     MOVE 11111 TO NR1
     MOVE 22222 TO NR2
     MOVE 33333 TO NR3
     CALL "MyFunV" USING R1
     STOP RUN.
```

Tareas relacionadas
“Pasar datos entre COBOL y C/C++” en la página 465

Referencias relacionadas
sentencia CALL (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: programas C a los que llama y llama COBOL

El ejemplo siguiente ilustra que una función C llamada recibe argumentos en el orden en el que se han pasado en una sentencia CALL de COBOL.

El archivo MyFun.c contiene el siguiente código fuente, que llama al programa COBOL tprog1:

```
#include <stdio.h>
extern void TPROG1(double *);
void
MyFun(short *ps1, short s2, long *k1, long *k2, double *m)
{
    double x;
    x = 2.0*(*m);
    printf("MyFun got s1=%d s2=%d k1=%d k2=%d x=%f\n",
             *ps1, s2, *k1,*k2, x);
}

long
MyFunR(short s1)
{
    return(s1 * s1);
}

double
MyFunD(double d1)
{
    double z;
    /* calling COBOL */
    z = 1122.3344;
    (void) TPROG1(&z);
    /* returning a value to COBOL */
    return(2.0 * d1);
}

void
MyFunV(long *pn)
{
    printf("MyFunV got %d %d %d\n", *pn, *(pn+1), *(pn+2));
}
```

MyFun.c consta de las funciones siguientes:

MiFun
Ilustra el paso de una variedad de argumentos.

MiFunR
Ilustra cómo pasar y devolver una variable larga.

MyFunD
Ilustra C que llama a un programa COBOL e ilustra cómo pasar y devolver una variable doble.

MiFunV
Ilustra el paso de un puntero a un registro y el acceso a los elementos del registro en un programa C.

Ejemplo: Programa COBOL llamado por un programa C

El ejemplo siguiente muestra cómo escribir programas COBOL a los que llaman los programas C.

La función C MiFund llama al programa COBOL tprog1 en el programa MyFun.c (consulte “Ejemplo: programas C a los que llama y llama COBOL” en la página 468). El programa COBOL llamado contiene el siguiente código fuente:

```
*
 IDENTIFICATION DIVISION.
 PROGRAM-ID. TPROG1.
*
 DATA DIVISION.
 LINKAGE SECTION.
*
 01 X                USAGE COMP-2.
*
 PROCEDURE DIVISION USING X.
     DISPLAY "TPROG1 got x= " X
     GOBACK.
```

Tareas relacionadas
“Llamada entre programas COBOL y C/C++” en la página 464

Ejemplo: resultados de la compilación y ejecución de ejemplos

Este ejemplo muestra cómo puede compilar, enlazar y ejecutar programas COBOL cobcallc.cbl y tprog.cbl y el programa C MyFun.c, y muestra los resultados de la ejecución de los programas.

Compile y enlace cobcallc.cbl, tprog.cbl y MyFun.c emitiendo los mandatos siguientes:

1. `gcc -m32 -c MyFun.c`
2. `cob2 cobcallc.cbl MyFun.o tprog1.cbl -o cobcallc`

Ejecute el programa emitiendo el mandato `cobcallc`. Los resultados son los siguientes:

```
call MyFun with n4=00123 ns4=-00567 n9=0098765432
            ns9=-0013579456 ns18=.222220000000000000E 03
MyFun got s1=123 s2=-567 k1=98765432 k2=-13579456 x=444.440000
n4=01024 and ns9= n4 times n4= 0001048576
TPROG1 got x=  .112233440000000000E 04
d1=-.35792568025000000E 09 and d2= 2.0 times d2= -.71585136050000000E 09
MyFunV got 11111 22222 33333
```

“Ejemplo: Programa COBOL que llama a funciones C” en la página 467
“Ejemplo: programas C a los que llama y llama COBOL” en la página 468
“Ejemplo: Programa COBOL llamado por un programa C” en la página 469

Ejemplo: Programa COBOL que llama a la función C++

El ejemplo siguiente muestra un programa COBOL que llama a una función C++ utilizando una sentencia CALL que pasa argumentos BY REFERENCE.

El ejemplo ilustra los conceptos siguientes:

- La sentencia CALL no indica si el programa llamado está escrito en COBOL o C++.
- Debe declarar un valor de retorno de función en una sentencia CALL que llame a una función C++ no vacía.
- Los tipos de datos COBOL deben correlacionarse con los tipos de datos C++ adecuados.
- La función C++ debe declararse `extern "C"`.
- Los argumentos COBOL se pasan BY REFERENCE. La función C++ los recibe utilizando parámetros de referencia.

Programa COBOL `driver`:

```
cbl pgmname(mixed)
Identification Division.
Program-Id. "driver".
Data division.
Working-storage section.
01 A pic 9(8) binary value 11111.
01 B pic 9(8) binary value 22222.
01 R pic 9(8) binary.
Procedure Division.
    Display "Hello World, from COBOL!"
    Call "sub" using by reference A B
      returning R
    Display R
    Stop Run.
```

Función C++ sub:

```
#include <iostream.h>
extern "C" long sub(long& A, long& B) {
  cout << "Hello from C++" << endl;
  return A + B;
}
```

Salida:

```
Hello World, from COBOL!
Hello from C++
00033333
```

Tareas relacionadas
"Pasar datos entre COBOL y C/C++" en la página 465

Referencias relacionadas
sentencia CALL (*COBOL for Linux en x86 Consulta de lenguaje*)

Realización de llamadas recursivas

Un programa llamado puede ejecutar directa o indirectamente su llamante. Por ejemplo, el programa X llama al programa Y, el programa Y llama al programa Z y, a continuación, el programa Z llama al programa X. Este tipo de llamada es *recursiva*.

Acerca de esta tarea

Para realizar una llamada recursiva, debe la cláusula `RECURSIVE` en el párrafo `PROGRAM-ID` del programa llamado recursivamente. Si intenta llamar de forma recursiva a un programa COBOL que no tiene la cláusula `RECURSIVE` en el párrafo `PROGRAM-ID` , la unidad de ejecución finalizará de forma anómala.

Tareas relacionadas
"Identificación de un programa como recursivo" en la página 4

Referencias relacionadas
Párrafo PROGRAM-ID (*COBOL for Linux en x86 Consulta de lenguaje*)

Pasar códigos de retorno

Puede utilizar el registro especial `RETURN-CODE` para pasar información entre programas compilados por separado.

Acerca de esta tarea

Puede establecer RETURN-CODE en un programa llamado antes de volver al llamante y, a continuación, probar este valor devuelto en el programa de llamada. Esta técnica se utiliza normalmente para indicar el nivel de éxito del programa llamado. Por ejemplo, se puede utilizar un RETURN-CODE de cero para indicar que el programa llamado se ha ejecutado correctamente.

Terminación normal: cuando un programa principal finaliza normalmente, el valor de RETURN-CODE se pasa al sistema operativo como un código de retorno de usuario. Sin embargo, Linux restringe los valores de código de retorno de usuario a 0 a 255. Por lo tanto, si, por ejemplo, RETURN-CODE contiene 258 cuando finaliza el programa, Linux envuelve el valor dentro del rango soportado, lo que da como resultado un código de retorno de usuario de 2.

Excepción irrecuperable: Cuando un programa encuentra una excepción irrecuperable, el código de retorno del usuario se establece en 128 más el número de señal. Para un programa sin hebras, la unidad de ejecución se termina; para un programa con hebras, la hebra en la que se ejecuta el programa, no la unidad de ejecución, se termina.

Tareas relacionadas
“Pasar información de código de retorno” en la página 481

Referencias relacionadas
RETURN-CODE (*COBOL for Linux en x86 Consulta de lenguaje*)

Capítulo 23. compartir datos

Si una unidad de ejecución consta de varios programas compilados por separado que se llaman entre sí, los programas deben poder comunicarse entre sí. También suelen necesitar acceso a datos comunes.

Acerca de esta tarea

Esta información describe cómo puede escribir programas que comparten datos con otros programas. En esta información, un *subprograma* es cualquier programa llamado por otro programa.

Tareas relacionadas
"Utilización de datos de otro programa" en la página 13

"Pasar datos" en la página 473
"Codificación de la SECCIÓN DE ENLACE" en la página 476
"Codificación de PROCEDURE DIVISION para pasar argumentos" en la página 477
"Utilización de punteros de procedimiento y función" en la página 481
"Pasar información de código de retorno" en la página 481
"Compartición de datos utilizando la cláusula EXTERNAL" en la página 482
"Compartir archivos entre programas (archivos externos)" en la página 483
"Utilización de argumentos de línea de mandatos" en la página 486

Pasar datos

Puede elegir entre tres formas de pasar datos entre programas: `BY REFERENCE`, `BY CONTENT`o `BY VALUE`.

Acerca de esta tarea

`BY REFERENCE`
El subprograma hace referencia y procesa los elementos de datos en el almacenamiento del programa de llamada en lugar de trabajar en una copia de los datos. `BY REFERENCE` es el mecanismo de paso supuesto para un parámetro si no se especifica ni se implica ninguna de las tres formas para el parámetro.

`BY CONTENT`
El programa de llamada sólo pasa el contenido del *literal* o *identificador*. El programa llamado no puede cambiar el valor del *literal* o *identificador* en el programa de llamada, incluso si modifica el elemento de datos en el que ha recibido el *literal* o el *identificador*.

`BY VALUE`
El programa o método de llamada pasa el valor del *literal* o *identificador*, no una referencia al elemento de datos de envío. El programa llamado o el método invocado pueden cambiar el parámetro. Sin embargo, debido a que el subprograma o método sólo tiene acceso a una copia temporal del elemento de datos de envío, cualquier cambio no afecta al argumento del programa de llamada.

Determine cuál de estos métodos de paso de datos debe utilizar basándose en lo que desea que haga el programa con los datos.

Tabla 45. ***Métodos para pasar datos en la sentencia CALL***		
Código	**objetivo**	**Comentarios**
`CALL . . . BY REFERENCE` *identificador*	Para que la definición del argumento de la sentencia `CALL` en el programa de llamada y la definición del parámetro en el programa llamado compartan la misma memoria	Cualquier cambio realizado por el subprograma en el parámetro afecta al argumento del programa de llamada.
`CALL . . . BY REFERENCE ADDRESS OF` *identificador*	Para pasar la dirección de *identificador* a un programa llamado, donde *identificador* es un elemento del `LINKAGE SECTION`	Los cambios realizados por el subprograma en la dirección afectan a la dirección del programa de llamada.
`CALL . . . BY CONTENT ADDRESS OF` *identificador*	Para pasar una copia de la dirección del *identificador* a un programa llamado	Los cambios en la copia de la dirección no afectarán a la dirección del *identificador*, pero los cambios en el *identificador* utilizando la copia de la dirección provocarán cambios en el *identificador*.
`CALL . . . BY CONTENT` *identificador*	Para pasar una copia del identificador al subprograma	Los cambios realizados en el parámetro por el subprograma no afectarán al identificador del interlocutor.
`CALL . . . BY CONTENT` *literal*	Para pasar una copia de un valor literal a un programa llamado	
`CALL . . . BY CONTENT LENGTH OF` *identificador*	Para pasar una copia de la longitud de un elemento de datos	El programa de llamada pasa la longitud del *identificador* de su registro especial `LENGTH` .

Una combinación de `BY REFERENCE` y `BY CONTENT` como, por ejemplo:
`CALL 'ERRPROC'`
`  USING BY REFERENCE A`
`  BY CONTENT LENGTH OF A.`
	Para pasar un elemento de datos y una copia de su longitud a un subprograma	

`CALL . . . BY VALUE` *identificador*	Para pasar datos a un programa, como un programa C/C++ , que utiliza los convenios de enlace de parámetros `BY VALUE`	Una copia del identificador se pasa directamente en la lista de parámetros.
`CALL . . . BY VALUE` *literal*	Para pasar datos a un programa, como un programa C/C++ , que utiliza los convenios de enlace de parámetros `BY VALUE`	Una copia del literal se pasa directamente en la lista de parámetros.
`CALL . . . BY VALUE ADDRESS OF` *identificador*	Para pasar la dirección de *identificador* a un programa llamado. Esta es la forma recomendada de pasar datos a un programa C/C++ que espera un puntero a los datos.	Los cambios en la copia de la dirección no afectarán a la dirección del *identificador*, pero los cambios en el *identificador* utilizando la copia de la dirección provocarán cambios en el *identificador*.
`CALL . . . RETURNING`	Para llamar a una función C/C++ con un valor de retorno de función	

Tareas relacionadas
“Descripción de argumentos en el programa de llamada” en la página 475
“Descripción de parámetros en el programa llamado” en la página 475
“Prueba de argumentos OMITIDOS” en la página 476
“Especificando CALL... DEVOLVIENDO” en la página 482
“Compartición de datos utilizando la cláusula EXTERNAL” en la página 482
“Compartir archivos entre programas (archivos externos)” en la página 483

Referencias relacionadas
sentencia CALL (*COBOL for Linux en x86 Consulta de lenguaje*)
La frase USING (*COBOL for Linux en x86 Consulta de lenguaje*)

Descripción de argumentos en el programa de llamada

En el programa de llamada, describa los argumentos en DATA DIVISION de la misma forma que otros elementos de datos en DATA DIVISION.

Acerca de esta tarea

El almacenamiento para argumentos sólo se asigna en el programa más externo. Por ejemplo, el programa A llama al programa B, que llama al programa C. Los elementos de datos se asignan en el programa A. Se describen en LINKAGE SECTION de los programas B y C, haciendo que el único conjunto de datos esté disponible para los tres programas.

Si hace referencia a datos en un archivo, el archivo debe estar abierto cuando se haga referencia a los datos.

Codifique la frase USING de la sentencia CALL para pasar los argumentos. Si pasa un elemento de datos BY VALUE, debe ser un elemento elemental.

Tareas relacionadas
“Codificación de la SECCIÓN DE ENLACE” en la página 476
“Codificación de PROCEDURE DIVISION para pasar argumentos” en la página 477

Referencias relacionadas
La frase USING (*COBOL for Linux en x86 Consulta de lenguaje*)

Descripción de parámetros en el programa llamado

Debe saber qué datos se están pasando desde el programa de llamada y describirlos en el LINKAGE SECTION de cada programa al que el programa de llamada llama directa o indirectamente.

Acerca de esta tarea

Codifique la frase USING después de la cabecera PROCEDURE DIVISION para nombrar los parámetros que reciben los datos que se pasan desde el programa de llamada.

Cuando se pasan argumentos al subprograma BY REFERENCE, no es válido que el subprograma especifique ninguna relación entre sus parámetros y cualquier campo que no sea el que se pasa y se define en el programa principal. El subprograma no debe:

- Defina un parámetro para que sea mayor en número total de bytes que el argumento correspondiente.
- Utilice referencias de subíndice para hacer referencia a elementos más allá de los límites de las tablas que el programa de llamada pasa como argumentos.
- Utilice la modificación de referencia para acceder a los datos más allá de la longitud de los parámetros definidos.
- Manipule la dirección de un parámetro para acceder a otros elementos de datos definidos en el programa de llamada.

Si se infringe alguna de las reglas anteriores, es posible que se produzcan resultados inesperados.

Tareas relacionadas
“Codificación de la SECCIÓN DE ENLACE” en la página 476

Referencias relacionadas
La frase USING (*COBOL for Linux en x86 Consulta de lenguaje*)

Prueba de argumentos OMITIDOS

Puede especificar que uno o más argumentos BY REFERENCE no se pasen a un programa llamado codificando la palabra clave OMITTED en lugar de esos argumentos en la sentencia CALL .

Acerca de esta tarea

Por ejemplo, para omitir el segundo argumento al llamar al programa sub1, codifique esta sentencia:

```
Call 'sub1' Using PARM1, OMITTED, PARM3
```

Los argumentos de la frase USING de la sentencia CALL deben coincidir con los parámetros del programa llamado en número y posición.

En un programa llamado, puede probar si un argumento se ha pasado como OMITTED comparando la dirección del parámetro correspondiente con NULL. Por ejemplo:

```
Program-ID. sub1.
. . .
Procedure Division Using RPARM1, RPARM2, RPARM3.
    If Address Of RPARM2 = Null Then
        Display 'No 2nd argument was passed this time'
    Else
        Perform Process-Parm-2
    End-If
```

Referencias relacionadas
sentencia CALL (*COBOL for Linux en x86 Consulta de lenguaje*)
La frase USING (*COBOL for Linux en x86 Consulta de lenguaje*)

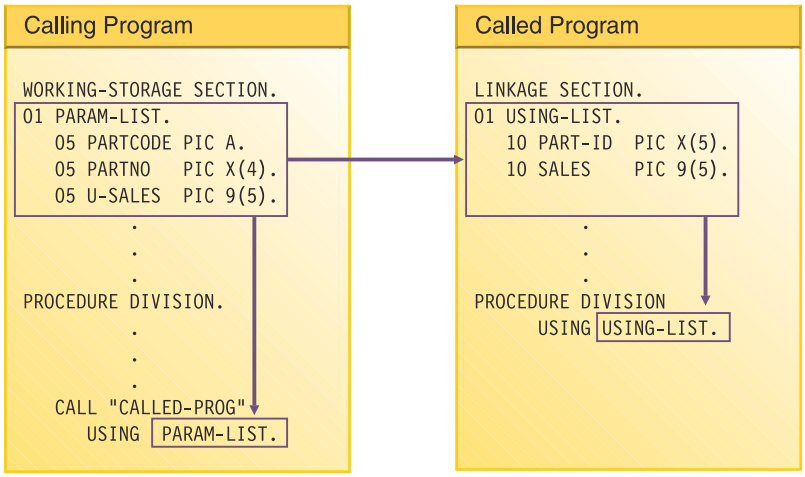
Codificación de la SECCIÓN DE ENLACE

Codifique el mismo número de nombres de datos en la lista de identificadores del programa llamado que el número de argumentos en el programa que realiza la llamada. Sincronizar por posición, porque el compilador pasa el primer argumento del programa de llamada al primer identificador del programa llamado, y así sucesivamente.

Acerca de esta tarea

Introducirá errores si el número de nombres de datos en la lista de identificadores de un programa llamado es mayor que el número de argumentos pasados desde el programa de llamada. El compilador no intenta coincidir con los argumentos y parámetros.

La figura siguiente muestra un elemento de datos que se pasa de un programa a otro (implícitamente BY REFERENCE):



En el programa de llamada, el código de las partes (PARTCODE) y el número de parte (PARTNO) son elementos de datos distintos. En el programa llamado, por el contrario, el código de las partes y el número de pieza se combinan en un elemento de datos (PART-ID). En el programa llamado, una referencia a PART-ID es la única referencia válida a estos elementos.

Codificación de PROCEDURE DIVISION para pasar argumentos

Si pasa un argumento BY VALUE, codifique la cláusula USING BY VALUE en la cabecera PROCEDURE DIVISION del subprograma. Si pasa un argumento BY REFERENCE o BY CONTENT, no es necesario que indique en la cabecera cómo se ha pasado el argumento.

Acerca de esta tarea

```
PROCEDURE DIVISION USING BY VALUE. . .
PROCEDURE DIVISION USING. . .
PROCEDURE DIVISION USING BY REFERENCE. . .
```

La primera cabecera anterior indica que los elementos de datos se pasan BY VALUE; la segunda o tercera cabecera indica que los elementos se pasan BY REFERENCE o BY CONTENT.

Referencias relacionadas
La cabecera de división de procedimiento (*COBOL for Linux en x86 Consulta de lenguaje*)
La frase USING (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia CALL (*COBOL for Linux en x86 Consulta de lenguaje*)

Agrupación de datos que se van a pasar

Acerca de esta tarea

Considere la posibilidad de agrupar todos los elementos de datos que necesita pasar entre programas y colocarlos bajo un elemento level-01 . Si lo hace, puede pasar un único registro level-01 .

Tenga en cuenta que si pasa un elemento de datos BY VALUE, debe ser un elemento elemental.

Para reducir la posibilidad de registros no coincidentes, coloque el registro level-01 en una biblioteca de copia y cópielo en ambos programas. Es decir, cópielo en el WORKING-STORAGE SECTION del programa de llamada y en el LINKAGE SECTION del programa llamado.

Tareas relacionadas
“Codificación de la SECCIÓN DE ENLACE” en la página 476

Referencias relacionadas
sentencia CALL (*COBOL for Linux en x86 Consulta de lenguaje*)

Manejo de series terminadas en nulo

COBOL da soporte a series terminadas en nulo cuando se utilizan sentencias de manejo de series junto con literales terminados en nulo y el literal hexadecimal X'00'.

Acerca de esta tarea

Puede manipular series terminadas en nulo (pasadas desde un programa C, por ejemplo) utilizando mecanismos de manejo de series como los del código siguiente:

```
01 L        pic X(20) value z'ab'.
01 M        pic X(20) value z'cd'.
01 N        pic X(20).
01 N-Length pic 99    value zero.
01 Y        pic X(13) value 'Hello, World!'.
```

Para determinar la longitud de una serie terminada en nulo y visualizar el valor de la serie y su longitud, codifique:

```
Inspect N tallying N-length for characters before initial X'00'
Display 'N: ' N(1:N-length) ' Length: ' N-length
```

Para mover una serie terminada en nulo a una serie alfanumérica, pero suprimir el código nulo:

```
Unstring N  delimited by X'00' into X
```

Para crear una serie terminada en nulo, codifique:

```
String Y      delimited by size
       X'00'  delimited by size
       into N.
```

Para concatenar dos series terminadas en nulo, codifique:

```
String L      delimited by x'00'
       M      delimited by x'00'
       X'00'  delimited by size
       into N.
```

Tareas relacionadas
“Manipulación de series terminadas en nulo” en la página 100

Referencias relacionadas
Literales alfanuméricos terminados en nulo
(*COBOL for Linux en x86 Consulta de lenguaje*)

Utilización de punteros para procesar una lista encadenada

Cuando necesite pasar y recibir direcciones de áreas de registro, puede utilizar elementos de datos de puntero, que son elementos de datos definidos con la cláusula USAGE IS POINTER o son registros especiales de ADDRESS OF .

Acerca de esta tarea

Una aplicación típica para utilizar elementos de datos de puntero está procesando una *lista encadenada*, una serie de registros en los que cada registro apunta al siguiente.

Cuando pasa direcciones entre programas en una lista encadenada, puede utilizar NULL para asignar el valor de una dirección que no es válida (0 no numérico) a un elemento de puntero de una de estas dos maneras:

- Utilice una cláusula `VALUE IS NULL` en su definición de datos.
- Utilice `NULL` como campo de envío en una sentencia SET .

En el caso de una lista encadenada en la que el elemento de datos de puntero del último registro contiene un valor nulo, puede utilizar este código para comprobar el final de la lista:

```
IF PTR-NEXT-REC = NULL
. . .
   (logic for end of chain)
```

Si el programa no ha llegado al final de la lista, el programa puede procesar el registro y pasar al siguiente registro.

Los datos pasados desde un programa de llamada pueden contener información de cabecera que desea ignorar. Puesto que los elementos de datos de puntero no son numéricos, no puede realizar directamente la aritmética en ellos. Sin embargo, para omitir la información de cabecera, puede utilizar la sentencia SET para incrementar la dirección pasada.

"Ejemplo: utilización de punteros para procesar una lista encadenada" en la página 479

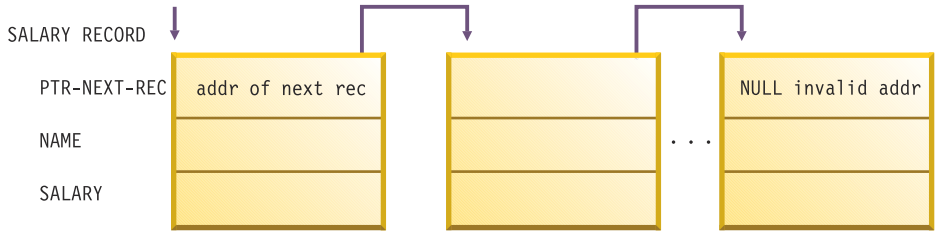
Tareas relacionadas
"Codificación de la SECCIÓN DE ENLACE" en la página 476
"Codificación de PROCEDURE DIVISION para pasar argumentos" en la página 477

Referencias relacionadas
"ADDR" en la página 268
sentencia SET (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: utilización de punteros para procesar una lista encadenada

El ejemplo siguiente muestra cómo puede procesar una lista enlazada, es decir, una lista encadenada de elementos de datos.

Para este ejemplo, muestra una lista encadenada de datos que consta de registros salariales individuales. La figura siguiente muestra una forma de visualizar cómo se enlazan los registros en el almacenamiento. El primer elemento de cada registro excepto el último apunta al siguiente registro. El primer elemento del último registro contiene un valor nulo (en lugar de una dirección válida) para indicar que es el último registro.



El pseudocódigo de alto nivel para una aplicación que procesa estos registros puede ser:

```
Obtain address of first record in chained list from routine
Check for end of the list
Do until end of the list
   Process record
   Traverse to the next record
End
```

El código siguiente contiene un esquema del programa de llamada, `LISTS`, utilizado en este ejemplo de proceso de una lista encadenada.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. LISTS.
 ENVIRONMENT DIVISION.
```

```
 DATA DIVISION.
******
 WORKING-STORAGE SECTION.
 77  PTR-FIRST         POINTER  VALUE IS NULL.                  (1)
 77  DEPT-TOTAL        PIC 9(4) VALUE IS 0.
******
 LINKAGE SECTION.
 01  SALARY-REC.
     02  PTR-NEXT-REC    POINTER.                               (2)
     02  NAME            PIC X(20).
     02  DEPT            PIC 9(4).
     02  SALARY          PIC 9(6).
 01  DEPT-X            PIC 9(4).
******
 PROCEDURE DIVISION USING DEPT-X.
******
* FOR EVERYONE IN THE DEPARTMENT RECEIVED AS DEPT-X,
* GO THROUGH ALL THE RECORDS IN THE CHAINED LIST BASED ON THE
* ADDRESS OBTAINED FROM THE PROGRAM CHAIN-ANCH
* AND ACCUMULATE THE SALARIES.
* IN EACH RECORD, PTR-NEXT-REC IS A POINTER TO THE NEXT RECORD
* IN THE LIST; IN THE LAST RECORD, PTR-NEXT-REC IS NULL.
* DISPLAY THE TOTAL.
******
     CALL "CHAIN-ANCH" USING PTR-FIRST                          (3)
     SET ADDRESS OF SALARY-REC TO PTR-FIRST                     (4)
******
     PERFORM WITH TEST BEFORE UNTIL ADDRESS OF SALARY-REC = NULL (5)

      IF DEPT = DEPT-X
        THEN ADD SALARY TO DEPT-TOTAL
        ELSE CONTINUE
      END-IF
      SET ADDRESS OF SALARY-REC TO PTR-NEXT-REC                 (6)

     END-PERFORM
******
     DISPLAY DEPT-TOTAL
     GOBACK.
```

(1)

PTR-FIRST se define como un elemento de datos de puntero con un valor inicial de NULL. En una devolución correcta de la llamada a CHAIN-ANCH, PTR-FIRST contiene la dirección del primer registro de la lista encadenada. Si algo va mal con la llamada, y PTR-FIRST nunca recibe el valor de la dirección del primer registro de la cadena, un valor nulo permanece en PTR-FIRST y, según la lógica del programa, los registros no se procesarán.

(2)

El LINKAGE SECTION del programa de llamada contiene la descripción de los registros de la lista encadenada. También contiene la descripción del código de departamento que se pasa en la cláusula USING de la sentencia CALL .

(3)

Para obtener la dirección del primer área de registro de SALARY-REC , el programa LISTS llama al programa CHAIN-ANCH.

(4)

La sentencia SET basa la descripción de registro SALARY-REC en la dirección contenida en PTR-FIRST.

(5)

La lista encadenada de este ejemplo se configura para que el último registro contenga una dirección que no sea válida. Esta comprobación del final de la lista encadenada se realiza con una estructura do-while donde el valor NULL se asigna al elemento de datos de puntero en el último registro.

(6)

La dirección del registro en LINKAGE-SECTION se establece igual a la dirección del siguiente registro mediante el elemento de datos de puntero enviado como primer campo en SALARY-REC. La rutina de proceso de registros se repite, procesando el siguiente registro de la lista encadenada.

Para incrementar las direcciones recibidas de otro programa, puede configurar LINKAGE SECTION y PROCEDURE DIVISION de este modo:

```
LINKAGE SECTION.
01  RECORD-A.
    02  HEADER          PIC X(12).
    02  REAL-SALARY-REC PIC X(30).
. . .
01  SALARY-REC.
    02  PTR-NEXT-REC    POINTER.
    02  NAME            PIC X(20).
    02  DEPT            PIC 9(4).
    02  SALARY          PIC 9(6).
. . .
PROCEDURE DIVISION USING DEPT-X.
. . .
    SET ADDRESS OF SALARY-REC TO ADDRESS OF REAL-SALARY-REC
```

La dirección de SALARY-REC se basa ahora en la dirección de REAL-SALARY-REC, o RECORD-A + 12.

Tareas relacionadas
“Utilización de punteros para procesar una lista encadenada” en la página 478

Utilización de punteros de procedimiento y función

Los *punteros de procedimiento* son elementos de datos definidos con la cláusula USAGE IS PROCEDURE-POINTER . Los *punteros de función* son elementos de datos definidos con la cláusula USAGE IS FUNCTION-POINTER .

Acerca de esta tarea

En esta información, "pointer" hace referencia a un elemento de datos de puntero de procedimiento o a un elemento de datos de puntero de función. Puede establecer cualquiera de estos elementos de datos para que contengan direcciones de entrada de, o punteros a, los siguientes puntos de entrada:

- Otro programa COBOL que no está anidado.
- Un programa escrito en otro idioma. Por ejemplo, para recibir la dirección de entrada de una función C, llame a la función con el formato CALL RETURNING de la sentencia CALL . Devuelve un puntero que puede convertir en un puntero de procedimiento utilizando un formato de la sentencia SET .
- Un punto de entrada alternativo en otro programa COBOL (tal como se define en una sentencia ENTRY).

Sólo puede establecer un elemento de datos de puntero utilizando la sentencia SET . Por ejemplo:

```
CALL 'MyCFunc' RETURNING ptr.
SET  proc-ptr  TO ptr.
CALL proc-ptr  USING dataname.
```

Supongamos que establece un elemento de puntero a una dirección de entrada en un módulo de carga al que llama una sentencia CALL *identifier* , y el programa cancela posteriormente ese módulo llamado. El elemento de puntero pasa a estar indefinido, y la referencia a él a partir de entonces no es fiable.

Referencias relacionadas
“ADDR” en la página 268

PROCEDURE-POINTER frase (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia SET (*COBOL for Linux en x86 Consulta de lenguaje*)

Pasar información de código de retorno

Utilice el registro especial RETURN-CODE para pasar códigos de retorno entre programas.

Acerca de esta tarea

Tareas relacionadas
"Pasar códigos de retorno" en la página 470

Utilización del registro especial RETURN-CODE

Acerca de esta tarea

Cuando un programa COBOL vuelve a su interlocutor, el contenido del registro especial RETURN-CODE se establece de acuerdo con el valor del registro especial RETURN-CODE en el programa llamado.

El establecimiento de RETURN-CODE mediante el programa llamado se limita a las llamadas entre programas COBOL. Por lo tanto, si el programa COBOL llama a un programa C, no puede esperar que se establezca el registro especial RETURN-CODE del programa COBOL.

Para una función equivalente entre programas COBOL y C, haga que el programa COBOL llame al programa C con la frase RETURNING . Si el programa C (función) declara correctamente un valor de función, se establecerá el valor RETURNING del programa COBOL de llamada.

Utilizando PROCEDURE DIVISION REGRESE...

Utilice la frase RETURNING en la cabecera PROCEDURE DIVISION de un programa para devolver información al programa de llamada.

Acerca de esta tarea

```
PROCEDURE DIVISION RETURNING dataname2
```

Cuando el programa llamado en el ejemplo anterior devuelve correctamente a su llamante, el valor de *dataname2* se almacena en el identificador que se ha especificado en la frase RETURNING de la sentencia CALL :

```
CALL . . . RETURNING dataname2
```

Especificando CALL... DEVOLVIENDO

Puede especificar la frase RETURNING de la sentencia CALL para llamadas a funciones de C/C++ o a subrutinas COBOL.

Acerca de esta tarea

La frase RETURNING tiene el formato siguiente.

```
CALL . . . RETURNING dataname2
```

El valor de retorno del programa llamado se almacena en *dataname2*. Debe definir *dataname2* en el DATA DIVISION del programa de llamada. El tipo de datos del valor de retorno declarado en la función de destino debe ser idéntico al tipo de datos de *dataname2*.

Compartición de datos utilizando la cláusula EXTERNAL

Utilice la cláusula EXTERNAL para habilitar programas compilados por separado (incluidos los programas en una secuencia por lotes) para compartir elementos de datos. Codifique EXTERNAL en la descripción de datos level-01 en WORKING-STORAGE SECTION.

Acerca de esta tarea

Se aplican las reglas siguientes:

- Los elementos que están subordinados a un EXTERNAL elemento de grupo son ellos mismos EXTERNAL.
- No puede utilizar el nombre de un elemento de datos EXTERNAL como nombre para otro elemento EXTERNAL en el mismo programa.
- No puede codificar la cláusula VALUE para cualquier elemento elemental , elemento de grupo o elemento subordinado que sea EXTERNAL.

En la unidad de ejecución, cualquier programa COBOL que tenga la misma descripción de datos para el elemento que el programa que contiene el elemento puede acceder y procesar dicho elemento. Por ejemplo, supongamos que el programa A tiene la siguiente descripción de datos:

```
01 EXT-ITEM1     EXTERNAL     PIC 99.
```

El programa B puede acceder a ese elemento de datos si B tiene la descripción de datos idéntica en su WORKING-STORAGE SECTION.

Cualquier programa que tenga acceso a un elemento de datos EXTERNAL puede cambiar el valor de dicho elemento. Por lo tanto, no utilice esta cláusula para los elementos de datos que necesita proteger.

Referencias relacionadas

Compartir archivos entre programas (archivos externos)

Para habilitar programas compilados por separado en una unidad de ejecución para acceder a un archivo como un archivo común, utilice la cláusula EXTERNAL para el archivo.

Acerca de esta tarea

Se recomienda que siga estas directrices:

- Utilice el mismo nombre de datos en la cláusula FILE STATUS de todos los programas que comprueban el código de estado de archivo.
- Para cada programa que compruebe el mismo campo de estado de archivo, codifique la cláusula EXTERNAL en la definición de datos level-01 para el campo de estado de archivo.

El uso de un archivo externo tiene estas ventajas:

- Incluso si el programa principal no contiene ninguna sentencia de entrada o salida, puede hacer referencia al área de registro del archivo.
- Cada subprograma puede controlar una sola función de entrada o salida, como OPEN o READ.
- Cada programa tiene acceso al archivo.

“Ejemplo: utilización de archivos externos” en la página 483

Tareas relacionadas
“Utilización de datos en operaciones de entrada y salida” en la página 10

Referencias relacionadas
Cláusula EXTERNAL (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: utilización de archivos externos

El ejemplo siguiente muestra el uso de un archivo externo en varios programas. Las sentencias COPY aseguran que cada subprograma contiene una descripción idéntica del archivo.

La tabla siguiente describe el programa principal y los subprogramas.

Name	Función
ef1	El programa principal, que llama a todos los subprogramas y, a continuación, verifica el contenido de un área de registro
ef1openo	Abre el archivo externo para la salida y comprueba el código de estado del archivo
ef1write	Escribe un registro en el archivo externo y comprueba el código de estado del archivo
ef1openi	Abre el archivo externo para entrada y comprueba el código de estado del archivo
ef1read	Lee un registro del archivo externo y comprueba el código de estado del archivo
ef1close	Cierra el archivo externo y comprueba el código de estado del archivo

Cada programa utiliza tres libros de copias:

- `efselect` se coloca en el párrafo `FILE-CONTROL` :

```
Select ef1
Assign To ef1
File Status Is efs1
Organization Is Sequential.
```

- `effile` se coloca en `FILE SECTION`:

```
Fd ef1 Is External
          Record Contains 80 Characters
          Recording Mode F.
01  ef-record-1.
    02 ef-item-1  Pic X(80).
```

- `efwrkstg` se coloca en `WORKING-STORAGE SECTION`:

```
01  efs1          Pic 99 External.
```

Entrada/salida utilizando archivos externos

```
 IDENTIFICATION DIVISION.
 Program-Id.
     ef1.
*
* This main program controls external file processing.
*
 ENVIRONMENT DIVISION.
 Input-Output Section.
 File-Control.
     Copy efselect.
 DATA DIVISION.
 FILE SECTION.
     Copy effile.
 WORKING-STORAGE SECTION.
     Copy efwrkstg.
 PROCEDURE DIVISION.
     Call "ef1openo"
     Call "ef1write"
     Call "ef1close"
     Call "ef1openi"
     Call "ef1read"
     If ef-record-1 = "First record" Then
       Display "First record correct"
     Else
       Display "First record incorrect"
       Display "Expected: " "First record"
       Display "Found   : " ef-record-1
     End-If
     Call "ef1close"
     Goback.
 End Program ef1.
 IDENTIFICATION DIVISION.
```

```
 Program-Id.
     ef1openo.
*
* This program opens the external file for output.
*
 ENVIRONMENT DIVISION.
 Input-Output Section.
 File-Control.
     Copy efselect.
 DATA DIVISION.
 FILE SECTION.
     Copy effile.
 WORKING-STORAGE SECTION.
     Copy efwrkstg.
 PROCEDURE DIVISION.
     Open Output ef1
     If efs1 Not = 0
       Display "file status " efs1 " on open output"
       Stop Run
     End-If
     Goback.
 End Program ef1openo.
 IDENTIFICATION DIVISION.
 Program-Id.
     ef1write.
*
* This program writes a record to the external file.
*
 ENVIRONMENT DIVISION.
 Input-Output Section.
 File-Control.
     Copy efselect.
 DATA DIVISION.
 FILE SECTION.
     Copy effile.
 WORKING-STORAGE SECTION.
     Copy efwrkstg.
 PROCEDURE DIVISION.
     Move "First record" to ef-record-1
     Write ef-record-1
     If efs1 Not = 0
       Display "file status " efs1 " on write"
       Stop Run
     End-If
     Goback.
 End Program ef1write.
 Identification Division.
 Program-Id.
     ef1openi.
*
* This program opens the external file for input.
*
 ENVIRONMENT DIVISION.
 Input-Output Section.
 File-Control.
     Copy efselect.
 DATA DIVISION.
 FILE SECTION.
     Copy effile.
 WORKING-STORAGE SECTION.
     Copy efwrkstg.
 PROCEDURE DIVISION.
     Open Input ef1
     If efs1 Not = 0
       Display "file status " efs1 " on open input"
       Stop Run
     End-If
     Goback.
 End Program ef1openi.
 Identification Division.
 Program-Id.
     ef1read.
*
* This program reads a record from the external file.
*
 ENVIRONMENT DIVISION.
 Input-Output Section.
 File-Control.
     Copy efselect.
 DATA DIVISION.
 FILE SECTION.
     Copy effile.
```

```
 WORKING-STORAGE SECTION.
     Copy efwrkstg.
 PROCEDURE DIVISION.
     Read ef1
     If efs1 Not = 0
       Display "file status " efs1 " on read"
       Stop Run
     End-If
     Goback.
 End Program ef1read.
 Identification Division.
 Program-Id.
     ef1close.
*
* This program closes the external file.
*
 ENVIRONMENT DIVISION.
 Input-Output Section.
 File-Control.
     Copy efselect.
 DATA DIVISION.
 FILE SECTION.
     Copy effile.
 WORKING-STORAGE SECTION.
     Copy efwrkstg.
 PROCEDURE DIVISION.
     Close ef1
     If efs1 Not = 0
       Display "file status " efs1 " on close"
       Stop Run
     End-If
     Goback.
 End Program ef1close.
```

Utilización de argumentos de línea de mandatos

Puede pasar argumentos a un programa principal en la línea de mandatos. El sistema operativo llama a los programas principales con series terminadas en nulo que contienen los argumentos.

Acerca de esta tarea

Si los argumentos que se especifican son más cortos que los nombres de datos COBOL que los reciben, utilice la técnica para aislarlos que se muestra en la tarea relacionada siguiente sobre la manipulación de series terminadas en nulo.

La forma en que se tratan los argumentos depende de si se utiliza la opción `-host` del mandato `cob2` .

Si no especifica la opción `-host` , los argumentos de línea de mandatos se pasan en formato de datos nativo, y Linux llama a todos los programas principales con los argumentos siguientes:

- Número de argumentos de línea de mandatos más uno
- Puntero al nombre del programa
- Puntero al primer argumento
- Puntero al segundo argumento
- . . .
- Puntero al argumento *n*

Si especifica la opción `-host` , Linux llama a todos los programas principales con una serie de caracteres EBCDIC que tiene un prefijo de media palabra que contiene la longitud de la serie. Debe especificar los argumentos de línea de mandatos como una sola serie entre comillas (`"`). Para pasar un carácter de comillas en la serie, preceda la comilla con el carácter de escape de barra inclinada invertida (`\`).

“Ejemplo: argumentos de línea de mandatos sin la opción -host” en la página 487
“Ejemplo: argumentos de línea de mandatos con la opción -host” en la página 487

Tareas relacionadas
“Manipulación de series terminadas en nulo” en la página 100
“Manejo de series terminadas en nulo” en la página 478

Referencias relacionadas
“Opciones cob2” en la página 248

Ejemplo: argumentos de línea de mandatos sin la opción -host

Este ejemplo muestra cómo leer argumentos de línea de mandatos si no está utilizando la opción `-host` .

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. "targlinux".
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 LINKAGE SECTION.
 01  PARM-LEN PIC S9(9) COMP.
 01  OS-PARM.
     02 PARMPTR-TABLE OCCURS 1 TO 100 TIMES DEPENDING ON PARM-LEN.
        03 PARMPTR  POINTER.
 01 PARM-STRING PIC XX.
*
 PROCEDURE DIVISION USING BY VALUE PARM-LEN BY REFERENCE OS-PARM.
     display "parm-len=" parm-len
     SET ADDRESS OF PARM-STRING TO PARMPTR(2).
     display "parm-string= '" PARM-STRING "'";
     EVALUATE PARM-STRING
         when "01"  display "case one"
         when "02"  display "case two"
         when "95"  display "case ninety-five"
         when other display "case unknown"
     END-EVALUATE
     GOBACK.
```

Supongamos que compila y ejecuta el programa siguiente:

```
cob2 targlinux.cbl
a.out 95
```

El resultado es:

```
parm-len=000000002
parm-string= '95'
case ninety-five
```

Ejemplo: argumentos de línea de mandatos con la opción -host

Este ejemplo muestra cómo leer los argumentos de línea de mandatos si está utilizando la opción `-host`.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. "testarg".
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 linkage section.
 01  os-parm.
     05 parm-len         pic s999 comp.
     05 parm-string.
         10 parm-char    pic x occurs 0 to 100 times
                         depending on parm-len.
```

```
*
 PROCEDURE DIVISION using os-parm.
     display "parm-len=" parm-len
     display "parm-string='" parm-string "'"
     evaluate parm-string
       when "01"  display "case one"
       when "02"  display "case two"
       when "95"  display "case ninety-five"
       when other display "case unknown"
     end-evaluate
     GOBACK.
```

Supongamos que compila y ejecuta el programa como se indica a continuación:

```
cob2 -host testarg.cbl
a.out "95"
```

A continuación, la salida resultante es:

```
parm-len=002
parm-string='95'
case ninety-five
```

Capítulo 24. Utilización de bibliotecas compartidas

Las bibliotecas compartidas proporcionan un medio cómodo y eficiente de empaquetar aplicaciones, y se utilizan ampliamente en Linux.

Acerca de esta tarea

En COBOL, una *biblioteca compartida* es una colección de uno o más programas externos. Del mismo modo que puede compilar y enlazar varios programas COBOL en un único archivo ejecutable, puede enlazar uno o varios programas COBOL más externos compilados para crear una biblioteca compartida. Normalmente se utiliza una biblioteca compartida como una colección de funciones llamadas con frecuencia.

Aunque los programas más externos de una biblioteca compartida pueden contener programas anidados, los programas externos a una biblioteca compartida sólo pueden llamar a los programas más externos (conocidos como *puntos de entrada*) de la biblioteca compartida. Cada programa de una biblioteca compartida se conoce como *subprograma*.

Puede llamar a bibliotecas compartidas COBOL o a una combinación de bibliotecas compartidas COBOL y C/C++ .

"Ejemplo: creación de una biblioteca compartida de ejemplo" en la página 490

Conceptos relacionados

"Programas principales, subprogramas y llamadas" en la página 457
"Enlace estático frente a utilización de bibliotecas compartidas" en la página 489
"Cómo resuelve el enlazador las referencias a las bibliotecas compartidas" en la página 490

Enlace estático frente a utilización de bibliotecas compartidas

El *enlace estático* es el enlace de un programa de llamada y uno o varios programas llamados en un único módulo ejecutable. Cuando se carga el programa, el sistema operativo coloca en la memoria un único archivo que contiene el código ejecutable y los datos.

La ventaja principal del enlace estático es que puede utilizarlo para crear módulos ejecutables independientes y autocontenidos.

Sin embargo, el enlace estático tiene estas desventajas:

- Puesto que los programas externos se crean en el archivo ejecutable, el archivo ejecutable aumenta de tamaño.
- No puede cambiar el comportamiento del archivo ejecutable sin volver a compilarlo y volver a enlazar.
- Si más de un programa de llamada necesita acceder a los programas llamados, las copias duplicadas de los programas llamados deben cargarse en la memoria.

Para superar estas desventajas, puede utilizar bibliotecas compartidas:

- Puede crear uno o más subprogramas en una biblioteca compartida; y varios programas pueden llamar a los subprogramas que están en la biblioteca compartida. Dado que el código de biblioteca compartida está separado del de los programas de llamada, los programas de llamada pueden ser más pequeños.
- Puede cambiar los subprogramas que están en la biblioteca compartida sin tener que volver a compilar o enlazar los programas de llamada.
- Sólo es necesario que una única copia de la biblioteca compartida esté en la memoria.

Las bibliotecas compartidas suelen proporcionar funciones comunes que pueden ser utilizadas por varios programas. Por ejemplo, puede utilizar bibliotecas compartidas para implementar paquetes de subprogramas, subsistemas e interfaces en otros programas o para crear bibliotecas de clases orientadas a objetos.

“Ejemplo: creación de una biblioteca compartida de ejemplo” en la página 490

Tareas relacionadas
“Llamada a programas COBOL no anidados” en la página 462

Cómo resuelve el enlazador las referencias a las bibliotecas compartidas

Cuando compila un programa, el compilador genera un módulo de objeto para el código del programa. Si llama a algún subprograma (*funciones* en C/C++, *subrutinas* en otros lenguajes) que están en un módulo de objeto externo, el compilador añade una referencia de programa externo al módulo de objeto de destino.

Para resolver una referencia externa a una biblioteca compartida, el enlazador añade información al archivo ejecutable que indica al cargador dónde encontrar el código de biblioteca compartida cuando se carga el archivo ejecutable.

El enlazador no resuelve todas las referencias a bibliotecas compartidas realizadas por sentencias COBOL CALL . Si la opción de compilador DYNAM está en vigor, COBOL resuelve sentencias CALL *literales* cuando se ejecutan estas llamadas. Las llamadas de CALL *identificador* también se resuelven dinámicamente.

“Ejemplo: creación de una biblioteca compartida de ejemplo” en la página 490

Conceptos relacionados
“Enlace estático frente a utilización de bibliotecas compartidas” en la página 489

Tareas relacionadas
“Realización de llamadas dinámicas a bibliotecas compartidas en CICS” en la página 411

Referencias relacionadas
“Archivos de entrada y salida de enlazador” en la página 252
“DYNAM” en la página 281

Ejemplo: creación de una biblioteca compartida de ejemplo

El ejemplo siguiente muestra tres programas COBOL, uno de los cuales (alpha) llama a los otros dos (beta y gamma). El procedimiento después de los programas muestra cómo crear una biblioteca compartida que contiene los dos programas llamados, crear una biblioteca de archivado que contiene esa biblioteca compartida y compilar y enlazar el programa que llama en un módulo que accede a los programas llamados en la biblioteca de archivado.

Ejemplo 1: alpha.cbl

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. alpha.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 hello1   pic x(30)  value is "message from alpha to beta".
 01 hello2   pic x(30)  value is "message from alpha to gamma".
*
 PROCEDURE DIVISION.
     display "alpha begins"
     call "beta" using hello1
     display "alpha after beta"
     call "gamma" using hello2
     display "alpha after gamma"
     goback.
```

Ejemplo 2: beta.cbl

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. beta.
*
 ENVIRONMENT DIVISION.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 Linkage section.
 01 msg   pic x(30).
*
 PROCEDURE DIVISION using msg.
     DISPLAY "beta gets msg=" msg.
     goback.
```

Ejemplo 3: gamma.cbl

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. gamma.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 Linkage section.
 01 msg            pic x(30).
*
PROCEDURE DIVISION using msg.
    DISPLAY "gamma gets msg=" msg.
    goback.
```

Procedimiento

La forma más sencilla de combinar los tres programas es compilarlos y enlazarlos en un único módulo ejecutable utilizando el mandato siguiente:

```
cob2 -o m_abg alpha.cbl beta.cbl gamma.cbl
```

A continuación, puede ejecutar los programas emitiendo el mandato `m_abg`, que da como resultado la salida siguiente:

```
alpha begins
beta gets msg=message from alpha to beta
alpha after beta
gamma gets msg=message from alpha to gamma
alpha after gamma
```

En lugar de enlazar los programas en un único módulo ejecutable, puede colocar `beta` y `gamma` en una biblioteca compartida (denominada `sh_bg` en el procedimiento siguiente), y compile y enlace `alpha` en un módulo ejecutable que acceda a `beta` y `gamma` en la biblioteca compartida de . Para ello, siga estos pasos:

1. Cree un script de versión que especifique los símbolos que la biblioteca compartida debe exportar:

   ```
   {
   global:
        GAMMA;
        BETA;
   local:
        *;
   };
   ```

Los nombres de símbolo del archivo de exportación `bg.version` que se muestran anteriormente están en mayúsculas porque el valor predeterminado de COBOL es utilizar nombres en mayúsculas para símbolos externos. Si necesita nombres en mayúsculas y minúsculas, utilice la opción de compilador `PGMNAME(MIXED)` .

Si denomina el archivo de exportación `bg.version`, debe utilizar la opción `-Wl,--version-script,bg.version` al crear la biblioteca compartida (tal como se muestra en el paso siguiente).

2. Utilice el mandato siguiente para combinar `beta` y `gamma` en un objeto de biblioteca compartida denominado `sh_bg`:

   ```
   cob2 -o sh_bg.so beta.cbl gamma.cbl -Wl,--version-script,bg.version
   ```

   Este mandato proporciona la siguiente información al compilador y al enlazador:

   - `-o sh_bg beta.cbl gamma.cbl` compila y enlaza `beta.cbl` y `gamma.cbl`, y nombra el módulo de salida resultante `sh_bg.so`.
   - `-Wl,--version-script,bg.version` indica al enlazador que exporte los símbolos que se denominan en el archivo de exportación `bg.version`.
3. Emita los mandatos siguientes para volver a compilar `alpha.cbl` y producir `m_alpha` ejecutables que tengan referencias externas resueltas en `sh_bg`:

   ```
   cob2 -o m_alpha alpha.cbl sh_bg.so
   ```

A continuación, puede ejecutar el programa emitiendo el mandato `m_alpha`, que genera la misma salida que la que se muestra antes de los pasos anteriores.

Tenga en cuenta que `sh_bg.so` debe estar en la variable de entorno LD_LIBRARY_PATH al ejecutar `m_alpha`. Por ejemplo, puede añadir el directorio actual a LD_LIBRARY_PATH emitiendo el mandato:

```
export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH
```

Como alternativa a emitir los mandatos en los dos últimos pasos, puede crear un archivo make que incluya los mandatos.

"Ejemplo: creación de un archivo make para la biblioteca compartida de ejemplo" en la página 492

Tareas relacionadas
"Pasar opciones al enlazador" en la página 251

Referencias relacionadas
"Opciones cob2" en la página 248
"Archivos de entrada y salida de enlazador" en la página 252
"PGMNAME" en la página 292

Ejemplo: creación de un archivo make para la biblioteca compartida de ejemplo

El ejemplo siguiente muestra cómo puede crear un archivo make para el objeto compartido , `sh_bg.so`, que contiene dos programas llamados, `beta` y `gamma`.

El archivo make presupone que el script de versión `bg.version` define los símbolos que exporta la biblioteca compartida.

(La creación de la biblioteca compartida se muestra en "Ejemplo: creación de una biblioteca compartida de ejemplo" en la página 490.)

```
#
#
all:     m_abg libbg.a m_alpha
```

```
#  Create m_abg containing alpha, beta, and gamma

m_abg:   alpha.cbl beta.cbl gamma.cbl
    cob2 -o m_abg alpha.cbl beta.cbl gamma.cbl

#  Create sh_bg.so containing beta and gamma
#  sh_bg.so is a shared object that exports the symbols defined in bg.version
#    contains both beta and gamma

sh_bg.so: beta.cbl gamma.cbl bg.version
    rm -f sh_bg.so
    cob2 -o sh_bg.so beta.cbl gamma.cbl -Wl,--version-script,bg.version

#  Create m_alpha containing alpha and using shared object sh_bg.so
m_alpha: alpha.cbl
    cob2 -o m_alpha alpha.cbl sh_bg.so

clean:
    rm -f m_abg m_alpha sh_bg.so *.lst
```

La ejecución del mandato `m_abg` o del mandato `m_alpha` proporciona la misma salida.

Capítulo 25. Preinicialización del entorno de ejecución COBOL

Utilizando la *preinicialización*, una aplicación puede inicializar el entorno de ejecución COBOL una vez, realizar varias ejecuciones utilizando dicho entorno y, a continuación, terminar explícitamente el entorno.

Puede utilizar la preinicialización para invocar programas COBOL varias veces desde un entorno no COBOL como, por ejemplo, C/C++.

La preinicialización tiene dos ventajas principales:

- El entorno COBOL permanece preparado para llamadas de programa.

  Puesto que una unidad de ejecución COBOL no se termina cuando se devuelve desde el primer programa COBOL de la unidad de ejecución, los programas COBOL que se invocan desde un entorno no COBOL se pueden invocar en su último estado utilizado.
- El rendimiento es mejor.

  La creación y desintegración repetidas veces del entorno de ejecución COBOL implica una sobrecarga y puede ralentizar una aplicación.

Utilice servicios de preinicialización para aplicaciones de varios lenguajes en las que los programas no COBOL necesitan utilizar un programa COBOL en su último estado utilizado. Por ejemplo, es posible que se abra un archivo en la primera llamada a un programa COBOL, y el programa invocador espera llamadas posteriores al programa para encontrar el archivo abierto.

Restricción: La preinicialización no está soportada en CICS.

Utilice las interfaces descritas en las tareas relacionadas para inicializar y terminar un entorno de ejecución COBOL persistente. Cualquier biblioteca compartida que contenga un programa COBOL utilizado en un entorno preinicializado no se puede suprimir hasta que termine el entorno preinicializado.

"Ejemplo: preinicialización del entorno COBOL" en la página 497

Tareas relacionadas

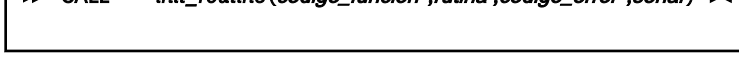
"Inicializando entorno COBOL persistente" en la página 495
"Terminando entorno COBOL preinicializado" en la página 496

Inicializando entorno COBOL persistente

Utilice la interfaz siguiente para inicializar un entorno COBOL persistente.

Acerca de esta tarea

Sintaxis de init_routine de CALL



CALL
Invocación de *init_routine*, utilizando elementos de lenguaje adecuados para el idioma desde el que se realiza la llamada

init_routine
El nombre de la rutina de inicialización: `_iwzCOBOLInit` o `IWZCOBOLINIT`

***código_función* (entrada)**
Un número binario de 4 bytes, pasado por valor. *código_función* puede ser:

1
El primer programa COBOL invocado después de esta invocación de función se trata como un subprograma.

***rutina* (entrada)**
Dirección de la rutina que se invocará si la unidad de ejecución termina. El argumento de señal pasado a esta función se pasa a la rutina de salida de terminación de unidad de ejecución. Esta rutina, cuando se invoca al terminar la unidad de ejecución, no debe volver al invocador de la rutina, sino utilizar longjmp () o exit ().

Si no proporciona una dirección de rutina de salida, se genera un *código_error* que indica que la preinicialización ha fallado.

***código_error* (salida)**
Un número binario de 4 bytes. *código_error* puede ser:

0
La preinicialización ha sido satisfactoria.

1
La preinicialización ha fallado.

***token* (entrada)**
Una señal de 4 bytes que se pasará a la *rutina* de salida especificada anteriormente cuando se invoque dicha rutina al terminar la unidad de ejecución.

Tareas relacionadas
“Terminando entorno COBOL preinicializado” en la página 496

Terminando entorno COBOL preinicializado

Utilice la interfaz siguiente para terminar el entorno COBOL persistente preinicializado.

Acerca de esta tarea

Sintaxis de CALL term_routine

CALL
Invocación de *term_routine*, utilizando elementos de lenguaje adecuados para el idioma desde el que se realiza la llamada

rutina-term_routine
El nombre de la rutina de terminación: `_iwzCOBOLTerm` o `IWZCOBOLTERM`

***código_función* (entrada)**
Un número binario de 4 bytes, pasado por valor. *código_función* puede ser:

1
Limpie el entorno de ejecución COBOL preinicializado como si se realizara una sentencia STOP RUN de COBOL; por ejemplo, todos los archivos COBOL están cerrados. Sin embargo, el control vuelve al llamante de este servicio.

***código_error* (salida)**
Un número binario de 4 bytes. *código_error* puede ser:

0
La terminación ha sido satisfactoria.

1
La terminación ha fallado.

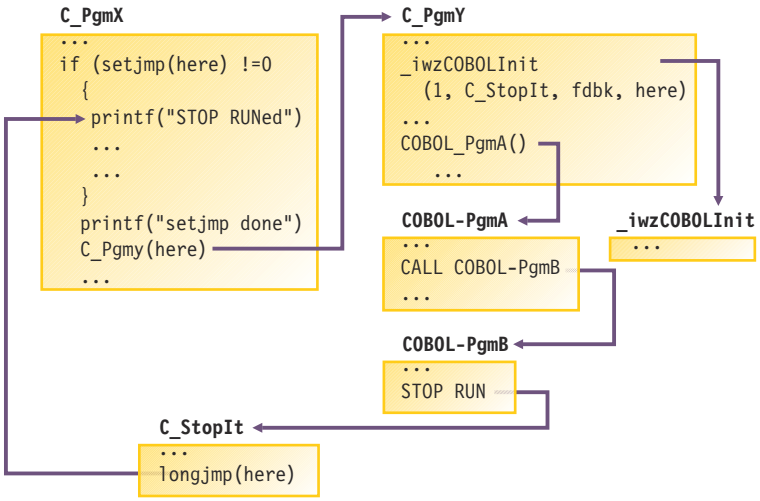
El primer programa COBOL llamado después de la invocación de la rutina de preinicialización se trata como un subprograma. Por lo tanto, un GOBACK de este programa (inicial) no desencadena la semántica de terminación de unidad de ejecución como, por ejemplo, el cierre de archivos. La terminación de unidad de ejecución (como con STOP RUN) libera el entorno COBOL preinicializado antes de la invocación de la rutina de salida de unidad de ejecución.

El entorno COBOL no está activo: si el programa invoca la rutina de terminación y el entorno COBOL todavía no está activo, la invocación no tiene ningún efecto en la ejecución y se devuelve el control al invocador con un código de error de 0.

"Ejemplo: preinicialización del entorno COBOL" en la página 497

Ejemplo: preinicialización del entorno COBOL

La figura siguiente ilustra cómo funciona el entorno COBOL preinicializado. El ejemplo muestra un programa C inicializando el entorno COBOL, llamando a programas COBOL y, a continuación, terminando el entorno COBOL.



El ejemplo siguiente muestra el uso de la preinicialización de COBOL. Un programa principal C llama al programa COBOL XIO varias veces. La primera llamada a XIO abre el archivo, la segunda llamada escribe un registro, etc. La llamada final cierra el archivo. A continuación, el programa C utiliza E/S de corriente C para abrir y leer el archivo.

Para probar y ejecutar el programa, especifique los mandatos siguientes desde un shell de mandatos:

```
xlc -c testinit.c
cob2 testinit.o xio.cbl
a.out
```

El resultado es:

```
_iwzCOBOLinit got 0
xio entered with x=0000000000
xio entered with x=0000000001
xio entered with x=0000000002
xio entered with x=0000000003
xio entered with x=0000000004
xio entered with x=0000000099
StopArg=0
_iwzCOBOLTerm expects rc=0 and got rc=0
FILE1 contains ----
11111
22222
33333
---- end of FILE1
```

Tenga en cuenta que en este ejemplo, la unidad de ejecución no ha sido terminada por un COBOL STOP RUN; se ha terminado cuando el programa principal llamó a `_iwzCOBOLTerm`.

El siguiente programa C está en el archivo `testinit.c`:

```
#ifdef _Linux
typedef int (*PFN)();
#define LINKAGE
#else
#include <windows.h>
#define LINKAGE _System
#endif

#include   <stdio.h>
#include   <setjmp.h>

extern void _iwzCOBOLInit(int fcode, PFN StopFun, int *err_code, void *StopArg);
extern void _iwzCOBOLTerm(int fcode, int *err_code);
extern void LINKAGE XIO(long *k);

jmp_buf Jmpbuf;
long StopArg = 0;

int LINKAGE
StopFun(long *stoparg)
{
        printf("inside StopFun\n");
        *stoparg = 123;
        longjmp(Jmpbuf,1);
}

main()
{
        int rc;
        long k;
        FILE *s;
        int c;

        if (setjmp(Jmpbuf) ==0) {
                _iwzCOBOLInit(1, StopFun, &rc, &StopArg);
                printf( "_iwzCOBOLinit got %d\n",rc);
                for (k=0; k <= 4; k++) XIO(&k);
                k = 99; XIO(&k);
        }
        else printf("return after STOP RUN\n");
        printf("StopArg=%d\n", StopArg);
        _iwzCOBOLTerm(1, &rc);
        printf("_iwzCOBOLTerm expects rc=0 and got rc=%d\n",rc);
        printf("FILE1 contains ---- \n");
        s = fopen("FILE1", "r");
        if (s) {
                while (  (c = fgetc(s) ) != EOF ) putchar(c);
        }
        printf("---- end of FILE1\n");
}
```

El siguiente programa COBOL está en el archivo `xio.cbl`:

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.     xio.
*****************************************************************
 ENVIRONMENT    DIVISION.
 CONFIGURATION   SECTION.
 INPUT-OUTPUT    SECTION.
 FILE-CONTROL.
     SELECT file1 ASSIGN TO FILE1
       ORGANIZATION IS LINE SEQUENTIAL
       FILE STATUS IS file1-status.
 . . .
 DATA           DIVISION.
 FILE SECTION.
 FD FILE1.
 01 file1-id pic x(5).
 . . .
 WORKING-STORAGE SECTION.
 01 file1-status  pic xx    value is zero.
 . . .
```

```
 LINKAGE SECTION.
*
 01 x               PIC S9(8) COMP-5.
 . . .
 PROCEDURE DIVISION using x.
 . . .
     display "xio entered with x=" x
     if x = 0 then
       OPEN output FILE1
     end-if
     if x = 1  then
        MOVE ALL "1" to file1-id
        WRITE file1-id
     end-if
     if x = 2 then
        MOVE ALL "2" to file1-id
        WRITE file1-id
     end-if
     if x = 3 then
        MOVE ALL "3" to file1-id
        WRITE file1-id
     end-if
     if x = 99 then
       CLOSE file1
     end-if
     GOBACK.
```

Capítulo 26. Procesando fechas de año de dos dígitos

Acerca de esta tarea

Con las extensiones de lenguaje de milenio (MLE), puede realizar cambios simples en los programas COBOL para definir campos de fecha. El compilador reconoce y actúa en estas fechas utilizando una ventana de siglo para garantizar la coherencia.

Utilice los pasos siguientes para implementar el reconocimiento automático de fechas en un programa COBOL:

Procedimiento

1. Añada la cláusula `DATE FORMAT` a las entradas de descripción de datos de los elementos de datos del programa que contienen fechas. Debe identificar todas las fechas con cláusulas `DATE FORMAT`, incluso las que no se utilizan en las comparaciones.
2. Para expandir las fechas, utilice las sentencias `MOVE` o `COMPUTE` para copiar el contenido de los campos de fecha con ventana en campos de fecha expandidos.
3. Si es necesario, utilice las funciones intrínsecas `DATEVAL` y `UNDATE` para convertir entre campos de fecha y no fechas.
4. Utilice la opción de compilador `YEARWINDOW` para establecer la ventana de siglo como una ventana fija o una ventana deslizante.
5. Compile el programa con la opción de compilador `DATEPROC(FLAG)` y revise los mensajes de diagnóstico para ver si el proceso de fecha ha producido algún efecto secundario inesperado.
6. Cuando la compilación sólo tiene mensajes de diagnóstico a nivel de información, puede volver a compilar el programa con la opción de compilador `DATEPROC(NOFLAG)` para producir un listado limpio.

Resultados

Puede utilizar determinadas técnicas de programación para aprovechar el proceso de fechas y controlar los efectos de utilizar campos de fecha como, por ejemplo, al comparar fechas, ordenar y fusionar por fecha y realizar operaciones aritméticas que implican fechas. Las extensiones de lenguaje del milenio soportan los campos de fecha de primer, único y último año para las operaciones más comunes en los campos de fecha: comparaciones, movimiento y almacenamiento, y aumento y disminución.

Conceptos relacionados
“Millennium Language Extensions (MLE)” en la página 502

Tareas relacionadas
“Resolución de problemas lógicos relacionados con la fecha” en la página 503
“Utilización de los campos de fecha de año primero, sólo año y último año” en la página 508
“Manipulación de literales como fechas” en la página 511
“Realización aritmética en campos de fecha” en la página 515
“Control explícito del proceso de fechas” en la página 517
“Analizar y evitar mensajes de diagnóstico relacionados con la fecha” en la página 519
“Evitar problemas en el proceso de fechas” en la página 520

Referencias relacionadas
“DATEPROC” en la página 277
“VENTANA” en la página 307
Cláusula DATE FORMAT (*COBOL for Linux en x86 Consulta de lenguaje*)

Millennium Language Extensions (MLE)

El término *extensiones de lenguaje de milenio* (MLE) hace referencia a las características de COBOL para Linux que la opción de compilador DATEPROC activa para ayudar con los problemas lógicos que implican fechas en el año 2000 y posteriores.

Cuando está habilitado, las extensiones incluyen:

- La cláusula DATE FORMAT . Para identificar los campos de fecha y especificar la ubicación del componente de año dentro de la fecha, añada esta cláusula a los elementos del DATA DIVISION.

  Hay varias restricciones sobre el uso de la cláusula DATE FORMAT ; por ejemplo, no puede especificarla para los elementos que tienen USAGE NATIONAL. Para obtener más detalles, consulte las referencias relacionadas a continuación.
- La reinterpretación como un campo de fecha del valor de retorno de la función para las siguientes funciones intrínsecas:
  - DATE-OF-INTEGER
  - DATE-TO-YYYYMMDD
  - DAY-OF-INTEGER
  - DAY-TO-YYYYDDD
  - YEAR-TO-YYYY
- La reinterpretación como un campo de fecha de los elementos de datos conceptuales DATE, DATE YYYYMMDD, DAYy DAY YYYYDDD en los siguientes formatos de la sentencia ACCEPT :
  - ACCEPT *identifier* FROM DATE
  - ACCEPT *identifier* FROM DATE YYYYMMDD
  - ACCEPT *identifier* FROM DAY
  - ACCEPT *identifier* FROM DAY YYYYDDD
- Las funciones intrínsecas UNDATE and DATEVAL, utilizadas para la reinterpretación selectiva de campos de fecha y no fechas.
- La función intrínseca YEARWINDOW, que recupera el año de inicio de la ventana de siglo establecida por la opción de compilador YEARWINDOW .

La opción de compilador DATEPROC habilita el proceso especial orientado a fechas de los campos de fecha identificados. La opción de compilador YEARWINDOW especifica la ventana de 100 años (la ventana de siglo) que se debe utilizar para interpretar los años con ventanas de dos dígitos.

Conceptos relacionados
“Principios y objetivos de estas prórrogas” en la página 502

Referencias relacionadas
“DATEPROC” en la página 277
“VENTANA” en la página 307
Restricciones en el uso de campos de fecha (*COBOL for Linux en x86 Consulta de lenguaje*)

Principios y objetivos de estas prórrogas

Para obtener el máximo beneficio de las extensiones de lenguaje del milenio, debe comprender las razones de su introducción en el lenguaje COBOL.

Las extensiones del milenio se centran en algunos principios clave:

- Los programas que se van a volver a compilar con semántica de fecha son activos totalmente probados y valiosos de la empresa. Su única limitación relevante es que los años de dos dígitos en los programas están restringidos al rango 1900-1999.
- No se realiza ningún proceso especial para la parte no anual de las fechas. Es por ello que la parte no anual de los formatos de fecha soportados se indica mediante Xs. De lo contrario, podría cambiar

el significado de los programas existentes. La única semántica sensible a la fecha que se proporciona implica expandir (y contraer) automáticamente la parte del año de dos dígitos de las fechas con respecto a la ventana del siglo para el programa.

- Las fechas con partes de año de cuatro dígitos son generalmente de interés sólo cuando se utilizan en combinación con fechas con ventanas. De lo contrario, hay poca diferencia entre las fechas de año de cuatro dígitos y las no fechas.

Sobre la base de estos principios, las extensiones lingüísticas del milenio están diseñadas para cumplir varios objetivos. Debe evaluar los objetivos que necesita cumplir para resolver sus problemas de procesamiento de fechas, y compararlos con los objetivos de las extensiones de lenguaje del milenio, para determinar cómo su aplicación puede beneficiarse de ellos. No debe considerar la posibilidad de utilizar las extensiones en aplicaciones nuevas o en mejoras en aplicaciones existentes, a menos que las aplicaciones estén utilizando datos antiguos que no se pueden expandir hasta más adelante.

Los objetivos de las ampliaciones del milenio son los siguientes:

- Amplíe la vida útil de los programas de aplicación tal como están especificados actualmente.
- Mantenga los cambios de origen en un mínimo, preferiblemente limitado a aumentar las declaraciones de campos de fecha en `DATA DIVISION`. Para implementar la solución de ventana de siglo, no es necesario cambiar la lógica del programa en `PROCEDURE DIVISION`.
- Conservar la semántica existente de los programas al añadir campos de fecha. Por ejemplo, cuando una fecha se expresa como un literal, como en la sentencia siguiente, el literal se considera compatible (con ventana o expandido) con el campo de fecha con el que se compara:

```
If Expiry-Date Greater Than 980101 . . .
```

  Dado que el programa existente presupone que las fechas de dos dígitos-año expresadas como literales están en el rango 1900-1999, las extensiones no cambian esta suposición.
- La característica de ventanas no está pensada para un uso a largo plazo. Puede ampliar la vida útil de las aplicaciones como un inicio hacia una solución a largo plazo que se puede implementar más adelante.
- La característica de campo de fecha expandida está pensada para un uso a largo plazo, como ayuda para expandir campos de fecha en archivos y bases de datos.

Las extensiones no proporcionan tipos de datos totalmente especificados o completos orientados a fecha, con semántica que reconozca, por ejemplo, las partes de mes y día de las fechas gregorianas. Sin embargo, proporcionan una semántica especial para la parte del año de las fechas.

Resolución de problemas lógicos relacionados con la fecha

Puede adoptar cualquiera de tres enfoques para ayudar con los problemas de procesamiento de fechas: utilizar una ventana de siglo, un puente interno o una expansión de campo completa.

Acerca de esta tarea

Ventana Siglo
Puede definir una ventana de siglo y especificar los campos que contienen fechas con ventanas. A continuación, el compilador interpreta los años de dos dígitos en estos campos de datos según la ventana del siglo.

Puente interno
Si sus archivos y bases de datos todavía no se han convertido a fechas de cuatro dígitos anuales, pero prefiere utilizar la lógica de cuatro dígitos de año ampliado en sus programas, puede utilizar una técnica de puente interno para procesar las fechas como fechas de cuatro dígitos anuales.

Expansión de campo completo
Esta solución implica expandir explícitamente los campos de fecha de año de dos dígitos para contener los años completos de cuatro dígitos en los archivos y bases de datos y, a continuación,

utilizar estos campos en formato expandido en los programas. Este es el único método que garantiza un proceso de fecha fiable para todas las aplicaciones.

Puede utilizar las extensiones de lenguaje del milenio con cada enfoque para lograr una solución, pero cada uno tiene ventajas y desventajas, como se muestra a continuación.

Tabla 46. ***Ventajas y desventajas de las soluciones del año 2000***

Aspecto	**Ventana de siglo**	**Puente interno**	**Expansión de campo completo**
Aplicación	Rápido y fácil, pero puede que no se adapte a todas las aplicaciones	Algún riesgo de dañar los datos	Debe asegurarse de que los cambios en bases de datos, libros de copias y programas estén sincronizados
Pruebas	Se necesitan menos pruebas porque no hay cambios en la lógica del programa	Las pruebas son fáciles porque los cambios en la lógica del programa son sencillos	
Duración del arreglo	Los programas pueden funcionar más allá de 2000, pero no una solución a largo plazo	Los programas pueden funcionar más allá de 2000, pero no una solución permanente	Solución permanente
Rendimiento	Puede degradar el rendimiento	Buen rendimiento	Mejor rendimiento
Mantenimiento			El mantenimiento es más fácil.

"Ejemplo: ventana de siglo" en la página 505
"Ejemplo: puente interno" en la página 506
"Ejemplo: conversión de archivos a formulario de fecha expandida" en la página 507

Tareas relacionadas

"Utilización de una ventana de siglo" en la página 504
"Utilización de puentes internos" en la página 505
"Pasando a expansión de campo completa" en la página 506

Utilización de una ventana de siglo

Una *ventana de siglo* es un intervalo de 100 años, como por ejemplo 1950-2049, dentro del cual cualquier año de dos dígitos es exclusivo. Para los campos de fecha con ventanas, especifique la fecha de inicio de la ventana de siglo utilizando la opción de compilador `YEARWINDOW` .

Acerca de esta tarea

Cuando la opción `DATEPROC` está en vigor, el compilador aplica esta ventana a los campos de fecha de dos dígitos del programa. Por ejemplo, dada la ventana de siglo 1930-2029, COBOL interpreta los años de dos dígitos como se indica a continuación:

- Los valores anuales de 00 a 29 se interpretan como años 2000-2029.
- Los valores de año de 30 a 99 se interpretan como años 1930-1999.

Para implementar esta ventana de siglo, utilice la cláusula `DATE FORMAT` para identificar los campos de fecha en el programa y utilice la opción de compilador `YEARWINDOW` para definir la ventana de siglo como una ventana fija o una ventana deslizante:

- Para una ventana fija, especifique un año de cuatro dígitos entre 1900 y 1999 como valor de la opción YEARWINDOW .

  Por ejemplo, YEARWINDOW(1950) define una ventana fija de 1950-2049.
- Para una ventana deslizante, especifique un entero negativo de -1 a -99 como valor de opción YEARWINDOW .

  Por ejemplo, YEARWINDOW(-50) define una ventana deslizante que empieza 50 años antes del año en el que se ejecuta el programa. Así que si el programa se está ejecutando en 2010, la ventana del siglo es 1960-2059; en 2011 la ventana del siglo se convierte automáticamente en 1961-2060, y así sucesivamente.

El compilador aplica automáticamente la ventana de siglo a las operaciones en los campos de fecha que ha identificado. No necesita ninguna lógica de programa adicional para implementar la ventana.

"Ejemplo: ventana de siglo" en la página 505

Referencias relacionadas
"DATEPROC" en la página 277
"VENTANA" en la página 307
Cláusula DATE FORMAT (*COBOL for Linux en x86 Consulta de lenguaje*)
Restricciones en el uso de campos de fecha (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: ventana de siglo

El ejemplo siguiente muestra (en negrita) cómo modificar un programa utilizando la cláusula DATE FORMAT para aprovechar la ventana de fecha automática.

```
CBLQUOTE,NOOPT,DATEPROC(FLAG),YEARWINDOW(-60)
. . .
01  Loan-Record.
    05  Member-Number   Pic X(8).
    05  DVD-ID          Pic X(8).
    05  Date-Due-Back   Pic X(6) Date Format yyxxxx.
    05  Date-Returned   Pic X(6) Date Format yyxxxx.
. . .
    If Date-Returned > Date-Due-Back Then
       Perform Fine-Member.
```

No hay cambios en PROCEDURE DIVISION. La adición de la cláusula DATE FORMAT a los dos campos de fecha significa que el compilador los reconoce como campos de fecha con ventanas y, por lo tanto, aplica la ventana de siglo al procesar la sentencia IF . Por ejemplo, si Date-Due-Back contiene 100102 (2 de enero de 2010) y Date-Returned contiene 091231 (31 de diciembre de 2009), Date-Returned es menor que (anterior a) Date-Due-Backy, por lo tanto, el programa no ejecuta el párrafo Fine-Member . (El programa comprueba si se ha devuelto un DVD a tiempo.)

Utilización de puentes internos

Para el puente interno, es necesario estructurar el programa de forma adecuada.

Acerca de esta tarea

Realice los pasos siguientes:

Procedimiento

1. Lea los archivos de entrada con fechas de año de dos dígitos.
2. Declare estas fechas de dos dígitos como campos de fecha con ventana y muévalas a campos de fecha expandidos, de modo que el compilador las expanda automáticamente a fechas de año de cuatro dígitos.
3. En el cuerpo principal del programa, utilice las fechas de año de cuatro dígitos para todo el proceso de fecha.

4. La ventana se remonta a años de dos dígitos.
5. Escriba las fechas de dos dígitos del año en los archivos de salida.

Resultados

Este proceso proporciona una ruta de migración conveniente a una solución de fecha ampliada completa, y puede tener ventajas de rendimiento sobre el uso de fechas con ventanas.

Cuando se utiliza esta técnica, los cambios en la lógica del programa son mínimos. Simplemente añada sentencias para expandir y contraer las fechas, y cambie las sentencias que hacen referencia a las fechas para utilizar los campos de fecha de año de cuatro dígitos en `WORKING-STORAGE` en lugar de los campos de año de dos dígitos en los registros.

Debido a que está convirtiendo las fechas de nuevo a años de dos dígitos para la salida, debe tener en cuenta la posibilidad de que el año está fuera de la ventana del siglo. Por ejemplo, si un campo de fecha contiene el año 2020, pero la ventana del siglo es 1920-2019, la fecha está fuera de la ventana. Simplemente mover el año a un campo de año de dos dígitos será incorrecto. Para protegerse contra este problema, puede utilizar una sentencia `COMPUTE` para almacenar la fecha, con la frase `ON SIZE ERROR` para detectar si la fecha está fuera de la ventana del siglo.

"Ejemplo: puente interno" en la página 506

Tareas relacionadas
"Utilización de una ventana de siglo" en la página 504
"Realización aritmética en campos de fecha" en la página 515
"Pasando a expansión de campo completa" en la página 506

Ejemplo: puente interno

El ejemplo siguiente muestra (en negrita) cómo se puede cambiar un programa para implementar el puente interno.

```
CBL  DATEPROC(FLAG),YEARWINDOW(-60)
    . . .
    File Section.
    FD  Customer-File.
    01  Cust-Record.
        05  Cust-Number     Pic 9(9) Binary.
        . . .
        05  Cust-Date       Pic 9(6) Date Format yyxxxx.
    Working-Storage Section.
    77  Exp-Cust-Date       Pic 9(8) Date Format yyyyxxxx.
    . . .
    Procedure Division.
        Open I-O Customer-File.
        Read Customer-File.
        Move Cust-Date to Exp-Cust-Date.
        . . .
   *=====================================================*
   * Use expanded date in the rest of the program logic  *
   *=====================================================*
        . . .
        Compute Cust-Date = Exp-Cust-Date
             On Size Error
                Display "Exp-Cust-Date outside century window"
        End-Compute
        Rewrite Cust-Record.
```

Pasando a expansión de campo completa

Utilizando las extensiones de lenguaje del milenio, puede avanzar gradualmente hacia una solución que amplíe completamente el campo de fecha.

Acerca de esta tarea

Realice los pasos siguientes:

Procedimiento

1. Aplique la solución de ventana de siglo y utilice esta solución hasta que tenga los recursos para implementar una solución más permanente.
2. Aplique la solución de puente interno. De esta manera puede utilizar fechas expandidas en sus programas mientras sus archivos siguen manteniendo fechas en formato de dos dígitos por año. Puede avanzar más fácilmente a una solución de expansión de campo completo porque no habrá más cambios en la lógica en el cuerpo principal de los programas.
3. Cambie los diseños de archivo y las definiciones de base de datos para que utilicen fechas de año de cuatro dígitos.
4. Cambie los libros de copias COBOL para que reflejen estos campos de fecha de año de cuatro dígitos.
5. Ejecute un programa de utilidad (o programa COBOL específico) para copiar archivos del formato antiguo al nuevo formato.
6. Vuelva a compilar los programas y realice pruebas de regresión y pruebas de fecha.

Resultados

Después de completar los dos primeros pasos, puede repetir los pasos restantes tantas veces como desee. No es necesario cambiar cada campo de fecha en cada archivo al mismo tiempo. Utilizando este método, puede seleccionar archivos para la conversión progresiva basándose en criterios como, por ejemplo, las necesidades empresariales o las interfaces con otras aplicaciones.

Cuando se utiliza este método, es necesario escribir programas especiales para convertir los archivos a formato de fecha ampliada.

“Ejemplo: conversión de archivos a formulario de fecha expandida” en la página 507

Ejemplo: conversión de archivos a formulario de fecha expandida

El ejemplo siguiente muestra un programa simple que copia de un archivo a otro mientras expande los campos de fecha. La longitud de registro del archivo de salida es mayor que la del archivo de entrada porque las fechas están expandidas.

```
CBL  QUOTE,NOOPT,DATEPROC(FLAG),YEARWINDOW(-80)
      ************************************************
      ** CONVERT - Read a file, convert the date    **
      **           fields to expanded form, write   **
      **           the expanded records to a new    **
      **           file.                            **
      ************************************************
       IDENTIFICATION DIVISION.
       PROGRAM-ID.  CONVERT.

       ENVIRONMENT DIVISION.

       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT INPUT-FILE
                  ASSIGN TO INFILE
                  FILE STATUS IS INPUT-FILE-STATUS.

           SELECT OUTPUT-FILE
                  ASSIGN TO OUTFILE
                  FILE STATUS IS OUTPUT-FILE-STATUS.

       DATA DIVISION.
       FILE SECTION.
       FD  INPUT-FILE
           RECORDING MODE IS F.
       01  INPUT-RECORD.
           03  CUST-NAME.
               05  FIRST-NAME  PIC X(10).
               05  LAST-NAME   PIC X(15).
           03  ACCOUNT-NUM     PIC 9(8).
           03  DUE-DATE        PIC X(6) DATE FORMAT YYXXXX.    (1)
           03  REMINDER-DATE   PIC X(6) DATE FORMAT YYXXXX.
           03  DUE-AMOUNT      PIC S9(5)V99 COMP-3.
```

```
FD  OUTPUT-FILE
    RECORDING MODE IS F.
01  OUTPUT-RECORD.
    03  CUST-NAME.
        05  FIRST-NAME  PIC X(10).
        05  LAST-NAME   PIC X(15).
    03  ACCOUNT-NUM     PIC 9(8).
    03  DUE-DATE        PIC X(8) DATE FORMAT YYYYXXXX.  (2)
    03  REMINDER-DATE   PIC X(8) DATE FORMAT YYYYXXXX.
    03  DUE-AMOUNT      PIC S9(5)V99 COMP-3.

WORKING-STORAGE SECTION.

01  INPUT-FILE-STATUS   PIC 99.
01  OUTPUT-FILE-STATUS  PIC 99.

PROCEDURE DIVISION.

    OPEN INPUT INPUT-FILE.
    OPEN OUTPUT OUTPUT-FILE.

READ-RECORD.
    READ INPUT-FILE
         AT END GO TO CLOSE-FILES.
    MOVE CORRESPONDING INPUT-RECORD TO OUTPUT-RECORD.   (3)
    WRITE OUTPUT-RECORD.

    GO TO READ-RECORD.

CLOSE-FILES.
    CLOSE INPUT-FILE.
    CLOSE OUTPUT-FILE.

    EXIT PROGRAM.

END PROGRAM CONVERT.
```

Notas:

(1)
Los campos DUE-DATE y REMINDER-DATE del registro de entrada son fechas gregorianas con componentes de año de dos dígitos. Se definen con una cláusula DATE FORMAT para que el compilador los reconozca como campos de fecha con ventanas.

(2)
El registro de salida contiene los mismos dos campos en formato de fecha expandida. Se definen con una cláusula DATE FORMAT para que el compilador los trate como campos de fecha de año de cuatro dígitos.

(3)
La sentencia MOVE CORRESPONDING mueve cada elemento en INPUT-RECORD a su elemento coincidente en OUTPUT-RECORD. Cuando los dos campos de fecha con ventana se mueven a los campos de fecha expandidos correspondientes, el compilador expande los valores de año utilizando la ventana del siglo actual.

Utilización de los campos de fecha de año primero, sólo año y último año

Cuando se comparan dos campos de fecha de los tipos de año primero o sólo de año, las dos fechas deben ser compatibles; es decir, deben tener el mismo número de caracteres no de año. No es necesario que el número de dígitos del componente de año sea el mismo.

Acerca de esta tarea

Un campo de fecha *año primero* es un campo de fecha cuya especificación DATE FORMAT consta de YY o YYYY, seguido de uno o más Xs. El formato de fecha de un campo de fecha de *sólo año* sólo tiene YY o YYYY. Un campo de fecha *year-last* es un campo de fecha cuya cláusula DATE FORMAT especifica uno o varios Xque preceden a YY o YYYY.

Los formatos de fecha de último año se suelen utilizar para mostrar fechas, pero son menos útiles computacionalmente porque el año, que es la parte más significativa de la fecha, está en la posición menos significativa de la representación de fecha.

Funcional para los campos de fecha de último año está limitado a comparaciones iguales o desiguales y determinados tipos de asignación. Los operandos deben ser fechas con formatos de fecha idénticos (año-último) o una fecha y una no fecha. El compilador no proporciona ventanas automáticas para operaciones en las últimas fechas del año. Cuando se produce un uso no soportado (por ejemplo, aritmética en las últimas fechas del año), el compilador proporciona un mensaje de nivel de error.

Si necesita más capacidad de proceso de fecha general para las fechas de último año, debe aislar y operar en la parte del año de la fecha.

"Ejemplo: comparación de campos de fecha de primer año" en la página 510

Conceptos relacionados
"Fechas compatibles" en la página 509

Tareas relacionadas
"Utilización de otros formatos de fecha" en la página 510

Fechas compatibles

El significado del término *fechas compatibles* depende de si el uso se produce en `DATA DIVISION` o en `PROCEDURE DIVISION`.

El uso de `DATA DIVISION` se ocupa de la declaración de los campos de fecha y de las reglas que rigen los elementos de lenguaje COBOL como, por ejemplo, los elementos de datos subordinados y la cláusula `REDEFINES` . En el ejemplo siguiente, `Review-Date` y `Review-Year` son compatibles porque `Review-Year` se puede declarar como un elemento de datos subordinado en `Review-Date`:

```
01  Review-Record.
    03  Review-Date              Date Format yyxxxx.
        05  Review-Year Pic XX    Date Format yy.
        05  Review-M-D  Pic XXXX.
```

El uso de `PROCEDURE DIVISION` se refiere a cómo se pueden utilizar los campos de fecha juntos en operaciones como comparaciones, movimientos y expresiones aritméticas. Para que los campos de fecha de primer año y sólo año se consideren compatibles, los campos de fecha deben tener el mismo número de caracteres no anuales. Por ejemplo, un campo con `DATE FORMAT YYXXXX` es compatible con otro campo que tiene el mismo formato de fecha y con un campo YYYYXXXX , pero no con un campo YYXXX .

Los campos de año-última fecha deben tener cláusulas `DATE FORMAT` idénticas. En concreto, no se permiten operaciones entre campos de fecha con ventanas y campos de fecha de último año expandido. Por ejemplo, puede mover un campo de fecha que tenga un formato de fecha de XXXXYY a otro campo de fecha de XXXXYY , pero no a un campo de fecha que tenga un formato de XXXXYYYY.

Puede realizar operaciones en campos de fecha, o en una combinación de campos de fecha y no fechas, siempre que los campos de fecha de la operación sean compatibles. Por ejemplo, supongamos las definiciones siguientes:

```
01  Date-Gregorian-Win  Pic 9(6) Packed-Decimal Date Format yyxxxx.
01  Date-Julian-Win     Pic 9(5) Packed-Decimal Date Format yyxxx.
01  Date-Gregorian-Exp  Pic 9(8) Packed-Decimal Date Format yyyyxxxx.
```

La sentencia siguiente es incoherente porque el número de dígitos que no son de año es diferente entre los dos campos:

```
If Date-Gregorian-Win Less than Date-Julian-Win . . .
```

La sentencia siguiente se acepta porque el número de dígitos no anuales es el mismo para ambos campos:

```
If Date-Gregorian-Win Less than Date-Gregorian-Exp . . .
```

En este caso, la ventana de siglo se aplica al campo de fecha con ventana (`Date-Gregorian-Win`) para asegurarse de que la comparación es significativa.

Cuando se utiliza una no fecha junto con un campo de fecha, se supone que la no fecha es compatible con el campo de fecha o se trata como un valor numérico simple.

Ejemplo: comparación de campos de fecha de primer año

El ejemplo siguiente muestra un campo de fecha con ventana que se compara con un campo de fecha expandido.

```
77  Todays-Date         Pic X(8) Date Format yyyyxxxx.
01  Loan-Record.
    05  Date-Due-Back   Pic X(6) Date Format yyxxxx.
. . .
    If Date-Due-Back > Todays-Date Then . . .
```

La ventana de siglo se aplica a `Date-Due-Back`. `Todays-Date` debe tener una cláusula `DATE FORMAT` para definirla como un campo de fecha expandido. Si no lo hiciera, se trataría como un campo sin fecha y, por lo tanto, se consideraría que tiene el mismo número de dígitos de año que `Date-Due-Back`. El compilador aplicaría la ventana de siglo asumida de 1900-1999, lo que crearía una comparación incoherente.

Utilización de otros formatos de fecha

Para ser elegible para la ventana automática, un campo de fecha debe contener un año de dos dígitos como la primera o única parte del campo. El resto del campo, si está presente, debe contener entre uno y cuatro caracteres, pero su contenido no es importante.

Acerca de esta tarea

Si hay campos de fecha en la aplicación que no se ajustan a estos criterios, es posible que tenga que realizar algunos cambios de código para definir sólo la parte del año de la fecha como un campo de fecha con la cláusula `DATE FORMAT` . Algunos ejemplos de estos tipos de formatos de fecha son:

- Campo de siete caracteres que consta de un año de dos dígitos, tres caracteres que contienen una abreviatura del mes y dos dígitos para el día del mes. Este formato no está soportado porque los campos de fecha sólo pueden tener de uno a cuatro caracteres no anuales.
- Una fecha gregoriana con el formato `DDMMYY`. La ventana automática no se proporciona porque el componente de año no es la primera parte de la fecha. Las fechas de último año como estas están totalmente soportadas como claves con ventanas en sentencias `SORT` o `MERGE` , y también están soportadas en un número limitado de otras operaciones COBOL.

Si necesita utilizar ventanas de fecha en casos como estos, tendrá que añadir un poco de código para aislar la parte del año de la fecha.

Ejemplo: aislar el año

El ejemplo siguiente muestra cómo puede aislar la parte del año de un campo de datos que tiene el formato `DDMMYY`.

```
03  Last-Review-Date Pic 9(6).
03  Next-Review-Date Pic 9(6).
```

```
. . .
Add 1 to Last-Review-Date Giving Next-Review-Date.
```

En el código anterior, si `Last-Review-Date` contiene 230110 (23 de enero de 2010), `Next-Review-Date` contendrá 230111 (23 de enero de 2011) después de que se ejecute la sentencia ADD . Se trata de un método sencillo para establecer la siguiente fecha para una revisión anual. Sin embargo, si `Last-Review-Date` contiene 230199, la adición de 1 produce 230200, que no es el resultado necesario .

Puesto que el año no es la primera parte de estos campos de fecha, la cláusula DATE FORMAT no se puede aplicar sin algún código para aislar el componente de año. En el ejemplo siguiente, el componente de año de ambos campos de fecha se ha aislado para que COBOL pueda aplicar la ventana de siglo y mantener resultados coherentes:

```
03  Last-Review-Date Date Format xxxxyy.
    05  Last-R-DDMM  Pic 9(4).
    05  Last-R-YY    Pic 99 Date Format yy.
03  Next-Review-Date Date Format xxxxyy.
    05  Next-R-DDMM  Pic 9(4).
    05  Next-R-YY    Pic 99 Date Format yy.
. . .
Move Last-R-DDMM to Next-R-DDMM.
Add 1 to Last-R-YY Giving Next-R-YY.
```

Manipulación de literales como fechas

Si un campo de fecha con ventanas tiene un nombre de condición level-88 asociado, el literal de la cláusula VALUE se crea con ventanas contra la ventana de siglo de la unidad de compilación en lugar de contra la ventana de siglo supuesta de 1900-1999.

Acerca de esta tarea

Por ejemplo, supongamos que tiene estas definiciones de datos:

```
05  Date-Due        Pic 9(6)  Date Format yyxxxx.
    88  Date-Target           Value 101220.
```

Si la ventana de siglo es 1950-2049, y el contenido de `Date-Due` es 101220 (que representa el 20 de diciembre de 2010), la primera condición siguiente se evalúa como verdadera, pero la segunda condición se evalúa como falsa:

```
If Date-Target. . .
If Date-Due = 101220
```

El literal `101220` se trata como un nondate; por lo tanto, está en ventana contra la ventana del siglo asumido de 1900-1999, y representa el 20 de diciembre de 1909. Pero cuando el literal se especifica en la cláusula VALUE de un nombre de condición level-88 , el literal pasa a formar parte del elemento de datos al que está adjunto. Puesto que este elemento de datos es un campo de fecha con ventana, la ventana de siglo se aplica siempre que se hace referencia a ese elemento de datos.

También puede utilizar la función intrínseca DATEVAL en una expresión de comparación para convertir un literal en un campo de fecha. El campo de fecha resultante se trata como un campo de fecha con ventana o un campo de fecha expandido para garantizar una comparación coherente. Por ejemplo, utilizando las definiciones anteriores, las dos condiciones siguientes se evalúan como verdaderas:

```
If Date-Due = Function DATEVAL (101220 "YYXXXX")
If Date-Due = Function DATEVAL (20101220 "YYYYXXXX")
```

Con un nombre de condición level-88 , puede especificar la opción THRU en la cláusula VALUE , pero debe especificar una ventana de siglo fijo en la opción de compilador YEARWINDOW en lugar de una ventana deslizante. Por ejemplo:

```
05  Year-Field  Pic 99  Date Format yy.
    88 In-Range         Value 98 Thru 06.
```

Con este formulario, el valor de ventana del segundo elemento del rango debe ser mayor que el valor de ventana del primer elemento. Sin embargo, el compilador sólo puede verificar esta diferencia si la opción de compilador YEARWINDOW especifica una ventana de siglo fija (por ejemplo, `YEARWINDOW(1940)` en lugar de `YEARWINDOW(-70)`).

El requisito de orden con ventana no se aplica a los campos de fecha de último año. Si especifica una cláusula VALUE de nombre de condición con la frase THROUGH para un campo de fecha de último año, los dos literales deben seguir las reglas normales de COBOL. Es decir, el primer literal debe ser menor que el segundo literal.

Conceptos relacionados
“Ventana de siglo asumido” en la página 512
“Tratamiento de no fechas” en la página 513

Tareas relacionadas
“Control explícito del proceso de fechas” en la página 517

Ventana de siglo asumido

Cuando un programa utiliza campos de fecha con ventanas, el compilador aplica la ventana de siglo definida por la opción de compilador YEARWINDOW a la unidad de compilación. Cuando se utiliza un campo de fecha con ventana junto con una no fecha, y el contexto exige que la no fecha se trate como una fecha con ventana, el compilador utiliza una ventana de siglo supuesta para resolver el campo de no fecha.

La ventana de siglo asumida es 1900-1999, que normalmente no es la misma que la ventana de siglo para la unidad de compilación.

En muchos casos, especialmente para las no fechas literales, esta ventana de siglo asumida es la opción correcta. En la siguiente construcción, el literal debe conservar su significado original del 1 de enero de 1972, y no cambiar a 2072 si la ventana del siglo es, por ejemplo, 1975-2074:

```
01  Manufacturing-Record.
    03  Makers-Date Pic X(6) Date Format yyxxxx.
. . .
    If Makers-Date Greater than "720101" . . .
```

Incluso si el supuesto es correcto, es mejor hacer que el año sea explícito y eliminar el mensaje de diagnóstico de nivel de aviso (que resulta de aplicar la ventana de siglo asumida) utilizando la función intrínseca DATEVAL :

```
If Makers-Date Greater than
    Function Dateval("19720101" "YYYYXXXX") . . .
```

En algunos casos, es posible que la suposición no sea correcta. En el ejemplo siguiente, supongamos que `Project-Controls` está en un miembro de copia utilizado por otras aplicaciones que todavía no se han actualizado para el proceso del año 2000 y, por lo tanto, `Date-Target` no puede tener una cláusula DATE FORMAT :

```
01  Project-Controls.
    03  Date-Target     Pic 9(6).
. . .
01  Progress-Record.
    03  Date-Complete   Pic 9(6) Date Format yyxxxx.
```

```
. . .
    If Date-Complete Less than Date-Target . . .
```

En el ejemplo anterior, es necesario que se cumplan las tres condiciones siguientes para que `Date-Complete` sea anterior a (menor que) `Date-Target`:

- La ventana del siglo es 1910-2009.
- `Date-Complete` es 991202 (fecha gregoriana: 2 de diciembre de 1999).
- `Date-Target` es 000115 (fecha gregoriana: 15 de enero de 2000).

Sin embargo, debido a que `Date-Target` no tiene una cláusula `DATE FORMAT` , no es una fecha. Por lo tanto, la ventana del siglo que se le aplica es la ventana del siglo asumido de 1900-1999, y se procesa como 15 de enero de 1900. Por lo tanto, `Date-Complete` será mayor que `Date-Target`, que no es el resultado necesario .

En este caso, debe utilizar la función intrínseca `DATEVAL` para convertir `Date-Target` en un campo de fecha para esta comparación. Por ejemplo:

```
If Date-Complete Less than
    Function Dateval (Date-Target "YYXXXX") . . .
```

Tareas relacionadas
“Control explícito del proceso de fechas” en la página 517

Tratamiento de no fechas

El modo en que el compilador trata una no fecha depende de su contexto.

Los elementos siguientes no son fechas:

- Un valor literal.
- Elemento de datos cuya descripción de datos no incluye una cláusula `DATE FORMAT` .
- Los resultados (intermedios o finales) de algunas expresiones aritméticas. Por ejemplo, la diferencia de dos campos de fecha es una no fecha, mientras que la suma de un campo de fecha y una no fecha es un campo de fecha.
- La salida de la función intrínseca `UNDATE` .

Cuando se utiliza una no fecha junto con un campo de fecha, el compilador interpreta la no fecha como una fecha cuyo formato es compatible con el campo de fecha o como un valor numérico simple. Esta interpretación depende del contexto en el que se utilizan el campo de fecha y la no fecha, como se indica a continuación:

- Comparación

  Cuando un campo de fecha se compara con una no fecha, la no fecha se considera compatible con el campo de fecha en el número de caracteres de año y no año. En el ejemplo siguiente, el literal no de fecha 971231 se compara con un campo de fecha con ventana:

  ```
  01  Date-1    Pic 9(6) Date Format yyxxxx.
  . . .
      If Date-1 Greater than 971231 . . .
  ```

  El literal no de fecha 971231 se trata como si tuviera el mismo `DATE FORMAT` que `Date-1`, pero con un año base de 1900.
- Operaciones aritméticas

En todas las operaciones aritméticas soportadas, los campos no de fecha se tratan como valores numéricos simples. En el ejemplo siguiente, el valor numérico 10000 se añade a la fecha gregoriana en `Date-2`, añadiendo de forma efectiva un año a la fecha:

```
01  Date-2    Pic 9(6) Date Format yyxxxx.
. . .
    Add 10000 to Date-2.
```

- Sentencia MOVE

  No se da soporte al movimiento de un campo de fecha a un campo que no es de fecha. Sin embargo, puede utilizar la función intrínseca UNDATE para hacerlo.

  Cuando mueve una fecha que no es de fecha a un campo de fecha, se presupone que el campo de envío es compatible con el campo de recepción en el número de caracteres de año y no de año. Por ejemplo, cuando mueve un campo de fecha sin fecha a una ventana, se supone que el campo sin fecha contiene una fecha compatible con un año de dos dígitos.

Utilización de condiciones de signo

Algunas aplicaciones utilizan valores especiales como, por ejemplo, ceros en los campos de fecha para actuar como un desencadenante, es decir, para indicar que se necesita algún proceso especial.

Acerca de esta tarea

Por ejemplo, en un archivo Pedidos, un valor de cero en `Order-Date` puede significar que el registro es un registro de totales de cliente en lugar de un registro de pedido. El programa compara la fecha con cero, como se indica a continuación:

```
01  Order-Record.
    05  Order-Date      Pic S9(5) Comp-3 Date Format yyxxx.
. . .
    If Order-Date Equal Zero Then . . .
```

Sin embargo, esta comparación no es válida porque el valor literal `Zero` no es una fecha y, por lo tanto, está en una ventana de siglo supuesta para dar un valor de 1900000.

De forma alternativa, puede utilizar una condición de signo en lugar de una comparación literal como se indica a continuación. Con una condición de signo, `Order-Date` se trata como una no fecha y la ventana de siglo no se tiene en cuenta.

```
If Order-Date Is Zero Then . . .
```

Este enfoque sólo se aplica si el operando de la condición de signo es un identificador simple en lugar de una expresión aritmética. Si se especifica una expresión, la expresión se evalúa en primer lugar, aplicándose la ventana de siglo cuando corresponda. A continuación, se compara la condición de signo con los resultados de la expresión.

Puede utilizar la función intrínseca UNDATE en su lugar para obtener el mismo resultado.

Conceptos relacionados
“Tratamiento de no fechas” en la página 513

Tareas relacionadas
“Control explícito del proceso de fechas” en la página 517

Referencias relacionadas
“DATEPROC” en la página 277

Realización aritmética en campos de fecha

Puede realizar operaciones aritméticas en campos de fecha numéricos de la misma forma que en cualquier elemento de datos numéricos. En su caso, se utilizará la ventana del siglo en el cálculo.

Acerca de esta tarea

Sin embargo, existen algunas restricciones sobre dónde se pueden utilizar los campos de fecha en expresiones aritméticas y sentencias. Las operaciones aritméticas que incluyen campos de fecha están restringidas a:

- Adición de una no fecha a un campo de fecha
- Restar una fecha no de un campo de fecha
- Restar un campo de fecha de un campo de fecha compatible para dar un resultado que no sea de fecha

Las siguientes operaciones aritméticas no están permitidas:

- Cualquier operación entre campos de fecha incompatibles
- Adición de dos campos de fecha
- Sustracción de un campo de fecha de un campo que no es de fecha
- Unario menos aplicado a un campo de fecha
- Multiplicación, división o exponenciación de o por un campo de fecha
- Expresiones aritméticas que especifican un campo de fecha de último año
- Expresiones aritméticas que especifican un campo de fecha de último año, excepto como elemento de datos de recepción cuando el campo de envío no es una fecha

La semántica de fecha se proporciona para las partes de año de los campos de fecha, pero no para las partes que no son de año. Por ejemplo, añadir 1 a un campo de fecha gregoriano con ventana que contiene el valor 980831 da un resultado de 980832, no de 980901.

Tareas relacionadas
“Cómo permitir el desbordamiento de campos de fecha con ventanas” en la página 515
“Especificación del orden de evaluación” en la página 516

Cómo permitir el desbordamiento de campos de fecha con ventanas

Un campo de fecha con ventanas (no anual) que participa en una operación aritmética se procesa como si el valor del componente de año del campo se incrementara por primera vez en 1900 o 2000, en función de la ventana del siglo.

Acerca de esta tarea

```
01 Review-Record.
    03 Last-Review-Year Pic 99 Date Format yy.
    03 Next-Review-Year Pic 99 Date Format yy.
. . .
    Add 10 to Last-Review-Year Giving Next-Review-Year.
```

En el ejemplo anterior, si la ventana de siglo es 1910-2009, y el valor de `Last-Review-Year` es 98, el cálculo continúa como si `Last-Review-Year` se incrementara por primera vez en 1900 para dar 1998. A continuación, se realiza la operación ADD , lo que proporciona un resultado de 2008. Este resultado se almacena en `Next-Review-Year` como 08.

Sin embargo, la siguiente declaración daría un resultado de 2018:

```
Add 20 to Last-Review-Year Giving Next-Review-Year.
```

Este resultado queda fuera del rango de la ventana del siglo. Si el resultado se almacena en `Next-Review-Year`, será incorrecto porque las referencias posteriores a `Next-Review-Year` lo interpretarán como 1918. En este caso, el resultado de la operación depende de si la frase `ON SIZE ERROR` se especifica en la sentencia `ADD` :

- Si se especifica `SIZE ERROR` , el campo receptor no se cambia y se ejecuta la sentencia imperativa `SIZE ERROR` .
- Si no se especifica `SIZE ERROR` , el resultado se almacena en el campo de recepción con los dígitos de la izquierda truncados.

Esta consideración es importante cuando se utiliza el puente interno. Cuando contrae un campo de fecha de año de cuatro dígitos de nuevo a dos dígitos para escribirlo en el archivo de salida, debe asegurarse de que la fecha se encuentre dentro de la ventana del siglo. A continuación, la fecha de año de dos dígitos se representará correctamente en el campo.

Para garantizar los cálculos adecuados, utilice una sentencia `COMPUTE` para realizar la contracción, con una frase `SIZE ERROR` para manejar la condición de fuera de ventana. Por ejemplo:

```
Compute Output-Date-YY = Work-Date-YYYY
On Size Error Perform CenturyWindowOverflow.
```

El proceso de `SIZE ERROR` para los receptores de fecha con ventana reconoce cualquier valor de año que esté fuera de la ventana de siglo. Es decir, un valor de año menor que el año inicial de la ventana de siglo genera la condición `SIZE ERROR` , al igual que un valor de año mayor que el año final de la ventana de siglo.

Tareas relacionadas
“Utilización de puentes internos” en la página 505

Especificación del orden de evaluación

Debido a las restricciones en los campos de fecha en las expresiones aritméticas, es posible que encuentre que los programas que se han compilado correctamente ahora producen mensajes de diagnóstico cuando algunos de los elementos de datos se cambian a campos de fecha.

Acerca de esta tarea

```
01 Dates-Record.
    03 Start-Year-1 Pic 99 Date Format yy.
    03 End-Year-1   Pic 99 Date Format yy.
    03 Start-Year-2 Pic 99 Date Format yy.
    03 End-Year-2   Pic 99 Date Format yy.
. . .
    Compute End-Year-2 = Start-Year-2 + End-Year-1 - Start-Year-1.
```

En el ejemplo anterior, la primera expresión aritmética evaluada es:

```
Start-Year-2 + End-Year-1
```

Sin embargo, la adición de dos campos de fecha no está permitida. Para resolver estos campos de fecha, debe utilizar paréntesis para aislar las partes de la expresión aritmética que están permitidas. Por ejemplo:

```
Compute End-Year-2 = Start-Year-2 + (End-Year-1 - Start-Year-1).
```

En este caso, la primera expresión aritmética evaluada es:

```
End-Year-1 - Start-Year-1
```

La resta de un campo de fecha de otro está permitida y da un resultado que no es de fecha. A continuación, este resultado no fecha se añade al campo de fecha `End-Year-1`, proporcionando un resultado de campo de fecha que se almacena en `End-Year-2`.

Control explícito del proceso de fechas

Puede haber ocasiones en las que desee que los elementos de datos COBOL se traten como campos de fecha sólo bajo determinadas condiciones o sólo en partes específicas del programa. O la aplicación puede contener campos de fecha de año de dos dígitos que no se pueden declarar como campos de fecha con ventana debido a alguna interacción con otro producto de software.

Acerca de esta tarea

Por ejemplo, si un campo de fecha se utiliza en un contexto en el que sólo es reconocido por su contenido binario verdadero sin interpretación adicional, la fecha de ese campo no se puede incluir en ventanas. Estos campos de fecha incluyen:

- Una clave en un archivo SdU
- Un campo de búsqueda en un sistema de base de datos como, por ejemplo, Db2
- Un campo de clave en un mandato CICS

Por el contrario, puede haber ocasiones en las que desee que un campo de fecha se trate como no fecha en partes específicas del programa.

COBOL proporciona dos funciones intrínsecas para tratar estas condiciones:

DATEVAL
Convierte una no fecha en un campo de fecha

UNDATE
Convierte un campo de fecha en un campo que no es de fecha

Tareas relacionadas
“Utilización de DATEVAL” en la página 517
“Utilización de UNDATE” en la página 518

Utilización de DATEVAL

Puede utilizar la función intrínseca DATEVAL para convertir una no fecha en un campo de fecha, de modo que COBOL aplicará el proceso de fecha relevante al campo.

Acerca de esta tarea

El primer argumento de la función es la no fecha que se va a convertir y el segundo argumento especifica el formato de fecha. El segundo argumento es una serie literal con una especificación similar a la del patrón de fecha en la cláusula `DATE FORMAT`.

En la mayoría de los casos, el compilador hace la suposición correcta sobre la interpretación de una no fecha pero acompaña a esta suposición con un mensaje de diagnóstico de nivel de aviso. Este mensaje suele aparecer cuando se compara una fecha de ventana con un literal:

```
03  When-Made        Pic x(6) Date Format yyxxxx.
. . .
If When-Made = "850701" Perform Warranty-Check.
```

Se asume que el literal es una fecha con ventana compatible pero con una ventana de siglo de 1900-1999, representando así el 15 de julio de 1985. Puede utilizar la función intrínseca DATEVAL para que el año de la fecha literal sea explícito y elimine el mensaje de aviso:

```
If When-Made = Function Dateval("19850701" "YYYYXXXX")
    Perform Warranty-Check.
```

"Ejemplo: DATEVAL" en la página 518

Utilización de UNDATE

Puede utilizar la función intrínseca UNDATE para convertir un campo de fecha en un campo no de fecha para que se pueda hacer referencia a él sin ningún proceso de fecha.

Acerca de esta tarea

Atención: Evite utilizar UNDATE excepto como último recurso, porque el compilador perderá el flujo de campos de fecha en el programa. Este problema podría hacer que las comparaciones de fechas no se efectuaran correctamente.

Utilice más cláusulas DATE FORMAT en lugar de la función UNDATE para MOVE y COMPUTE.

"Ejemplo: UNDATE" en la página 518

Ejemplo: DATEVAL

Este ejemplo muestra un caso en el que es mejor dejar un campo como no fecha y utilizar la función intrínseca DATEVAL en una sentencia de comparación.

Supongamos que se hace referencia a un campo Date-Copied muchas veces en un programa, pero que la mayoría de las referencias sólo mueven el valor entre registros o lo vuelven a formatear para su impresión. Sólo una referencia se basa en ella para contener una fecha (para la comparación con otra fecha). En este caso, es mejor dejar el campo como no fecha y utilizar la función intrínseca DATEVAL en la sentencia de comparación. Por ejemplo:

```
03  Date-Distributed Pic 9(6) Date Format yyxxxx.
03  Date-Copied      Pic 9(6).
. . .
If Function DATEVAL(Date-Copied "YYXXXX") Less than Date-Distributed . . .
```

En este ejemplo, DATEVAL convierte Date-Copied en un campo de fecha para que la comparación sea significativa.

Referencias relacionadas
DATEVAL (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: UNDATE

El ejemplo siguiente muestra un caso en el que es posible que desee convertir un campo de fecha en un campo no de fecha.

El campo Invoice-Date es una fecha juliana con ventana. En algunos registros, contiene el valor 00999 para indicar que el registro no es un registro de factura verdadero, sino que contiene información de control de archivos.

Invoice-Date tiene una cláusula DATE FORMAT porque la mayoría de sus referencias en el programa son específicas de la fecha. Sin embargo, cuando se comprueba la existencia de un registro de control, el valor 00 en el componente de año dará lugar a cierta confusión. Un valor de año de 00 en Invoice-Date podría representar 1900 o 2000, en función de la ventana del siglo. Esto se compara con un valor nondate (el literal 00999 en el ejemplo), que siempre se mostrará en la ventana del siglo asumido y, por lo tanto, siempre representa el año 1900.

Para garantizar una comparación coherente, debe utilizar la función intrínseca UNDATE para convertir `Invoice-Date` a una no fecha. Por lo tanto, si la sentencia IF no compara los campos de fecha, no es necesario que aplique ventanas. Por ejemplo:

```
01  Invoice-Record.
    03  Invoice-Date    Pic x(5) Date Format yyxxx.
. . .
    If FUNCTION UNDATE(Invoice-Date) Equal "00999" . . .
```

Referencias relacionadas
UNDATE (*COBOL for Linux en x86 Consulta de lenguaje*)

Analizar y evitar mensajes de diagnóstico relacionados con la fecha

Cuando la opción de compilador DATEPROC(FLAG) está en vigor, el compilador genera mensajes de diagnóstico para cada sentencia que define o hace referencia a un campo de fecha.

Acerca de esta tarea

Al igual que con todos los mensajes generados por el compilador, cada mensaje relacionado con la fecha tiene uno de los siguientes niveles de gravedad:

- Nivel de información, para llamar su atención sobre la definición o el uso de un campo de fecha.
- Nivel de aviso, para indicar que el compilador ha tenido que hacer una suposición sobre un campo de fecha o no fecha debido a información inadecuada codificada en el programa, o para indicar la ubicación de la lógica de fecha que se debe comprobar manualmente para comprobar si es correcta. La compilación procede, con cualquier supuesto que se siga aplicando.
- Nivel de error, para indicar que el uso del campo de fecha es incorrecto. La compilación continúa, pero los resultados de tiempo de ejecución son imprevisibles.
- Nivel de gravedad, para indicar que el uso del campo de fecha es incorrecto. La sentencia que ha causado este error se descarta de la compilación.

La forma más fácil de utilizar los mensajes MLE es compilar con un valor de opción FLAG que incluya los mensajes en el listado fuente después de la línea a la que hacen referencia los mensajes. Puede elegir ver todos los mensajes MLE o sólo determinadas gravedades.

Para ver todos los mensajes MLE, especifique las opciones de compilador FLAG(I,I) y DATEPROC(FLAG) . Inicialmente, es posible que desee ver todos los mensajes para comprender cómo MLE está procesando los campos de fecha en el programa. Por ejemplo, si desea realizar un análisis estático del uso de fecha en un programa utilizando el listado de compilación, utilice FLAG (I,I).

Sin embargo, se recomienda especificar FLAG(W,W) para compilaciones específicas de MLE. Debe resolver todos los mensajes de error de nivel grave (nivel S) y también debe resolver todos los mensajes de nivel de error (nivel E). Para los mensajes de nivel de aviso (nivel W), debe examinar cada mensaje y utilizar las directrices siguientes para eliminar el mensaje o, para los mensajes inevitables, asegurarse de que el compilador realiza suposiciones correctas:

- Los mensajes de diagnóstico pueden indicar algunos elementos de datos de fecha que deberían tener una cláusula DATE FORMAT . Añada cláusulas DATE FORMAT a estos elementos o utilice la función intrínseca DATEVAL en las referencias a ellos.
- Preste especial atención a los literales en condiciones de relación que implican campos de fecha o en expresiones aritméticas que incluyen campos de fecha. Puede utilizar la función DATEVAL en literales (así como elementos de datos que no son de fecha) para especificar un patrón de DATE FORMAT que se utilizará. Como último recurso, puede utilizar la función UNDATE para habilitar un campo de fecha que se utilizará en un contexto en el que no desea un comportamiento orientado a fechas.
- Con las cláusulas REDEFINES y RENAMES , el compilador puede producir un mensaje de diagnóstico de nivel de aviso si un campo de fecha y un valor nondate ocupan la misma ubicación de almacenamiento.

Debe comprobar estos casos detenidamente para confirmar que todos los usos de los elementos de datos con alias son correctos, y que ninguna de las redefiniciones percibidas que no son de fecha es realmente una fecha o puede afectar negativamente a la lógica de fecha en el programa.

En algunos casos, un mensaje de nivel W puede ser aceptable, pero es posible que desee cambiar el código para obtener una compilación con un código de retorno de cero.

Para evitar mensajes de diagnóstico de nivel de aviso, siga estas directrices:

- Añada cláusulas `DATE FORMAT` a los elementos de datos que contengan fechas, incluso si los elementos no se utilizan en comparaciones. Pero consulte las referencias relacionadas a continuación sobre las restricciones en el uso de campos de fecha. Por ejemplo, no puede utilizar la cláusula `DATE FORMAT` en un elemento de datos que se describe implícita o explícitamente como `USAGE NATIONAL`.
- No especifique un campo de fecha en un contexto en el que un campo de fecha no tenga sentido, como un elemento `FILE STATUS`, `PASSWORD`, `ASSIGN USING`, `LABEL RECORD`o `LINAGE` . Si lo hace, obtendrá un mensaje de nivel de aviso y el campo de fecha se tratará como no fecha.
- Asegúrese de que los alias implícitos o explícitos para los campos de fecha son compatibles, como por ejemplo en un elemento de grupo que consta únicamente de un campo de fecha.
- Asegúrese de que si un campo de fecha se define con una cláusula `VALUE` , el valor es compatible con la definición de campo de fecha.
- Utilice la función intrínseca `DATEVAL` si desea que una no fecha se trate como un campo de fecha, como por ejemplo al mover una no fecha a un campo de fecha o al comparar una fecha con ventana con una no fecha y desea una comparación de fecha con ventana. Si no utiliza `DATEVAL`, el compilador hará una suposición sobre el uso de la no fecha y generará un mensaje de diagnóstico de nivel de aviso. Incluso si la suposición es correcta, es posible que desee utilizar `DATEVAL` para eliminar el mensaje.
- Utilice la función intrínseca `UNDATE` si desea que un campo de fecha se trate como un campo sin fecha, como mover un campo de fecha a un campo sin fecha, o comparar un campo de fecha sin fecha y un campo de fecha con ventana cuando no desea una comparación con ventana.

Tareas relacionadas
“Control explícito del proceso de fechas” en la página 517
COBOL Millennium Language Extensions Guide (Análisis relacionado con la fecha mensajes de diagnóstico)

Referencias relacionadas
Restricciones en el uso de campos de fecha (*COBOL for Linux en x86 Consulta de lenguaje*)

Evitar problemas en el proceso de fechas

Cuando cambia un programa COBOL para utilizar las extensiones de lenguaje millennium, es posible que encuentre que algunas partes del programa necesitan una atención especial para resolver cambios imprevistos en el comportamiento. Por ejemplo, es posible que tenga que evitar problemas con campos decimales empaquetados y problemas que se producen si pasa de campos de fecha expandidos a campos de fecha con ventanas.

Acerca de esta tarea

Tareas relacionadas
“Evitar problemas con campos decimales empaquetados” en la página 520
“Pasar de campos de fecha expandidos a campos de fecha con ventanas” en la página 521

Evitar problemas con campos decimales empaquetados

Los campos `COMPUTATIONAL-3` (formato decimal empaquetado) a menudo se definen como que tienen un número impar de dígitos incluso si el campo no se utilizará para contener un número de esa magnitud. La representación interna de números decimales empaquetados siempre permite un número impar de dígitos.

Acerca de esta tarea

Un campo que contiene una fecha gregoriana de seis dígitos, por ejemplo, se puede declarar como `PIC S9(6) COMP-3`. Esta declaración reservará 4 bytes de almacenamiento. Pero un programador podría haber declarado el campo como `PIC S9(7)`, sabiendo que esto reservaría 4 bytes con el dígito de orden superior que siempre contiene un cero.

Si añade una cláusula `DATE FORMAT YYXXXX` a este campo, el compilador emitirá un mensaje de diagnóstico porque el número de dígitos de la cláusula `PICTURE` no coincide con el tamaño de la especificación de formato de fecha. En este caso, debe comprobar cuidadosamente cada uso del campo. Si el dígito de orden superior no se utiliza nunca, simplemente puede cambiar la definición de campo a `PIC S9(6)`. Si se utiliza (por ejemplo, si el mismo campo puede contener un valor que no sea una fecha), debe realizar alguna otra acción, como por ejemplo:

- Utilización de una cláusula `REDEFINES` para definir el campo como una fecha y una no fecha (este uso también generará un mensaje de diagnóstico de nivel de aviso)
- Definición de otro campo `WORKING-STORAGE` para mantener la fecha y mover el campo numérico al nuevo campo
- No añadir una cláusula `DATE FORMAT` al elemento de datos y utilizar la función intrínseca `DATEVAL` al hacer referencia al mismo como un campo de fecha

Pasar de campos de fecha expandidos a campos de fecha con ventanas

Cuando mueve un campo de fecha alfanumérico expandido a un campo de fecha con ventana, el movimiento no sigue los convenios COBOL normales para los movimientos alfanuméricos. Cuando los campos de envío y de recepción son campos de fecha, el movimiento se justifica a la derecha, no a la izquierda se justifica como normal. Para un movimiento expandido a ventana (contracción), los dos dígitos iniciales del año se truncan.

Acerca de esta tarea

En función del contenido del campo de envío, los resultados de este movimiento pueden ser incorrectos. Por ejemplo:

```
77  Year-Of-Birth-Exp  Pic x(4) Date Format yyyy.
77  Year-Of-Birth-Win  Pic xx   Date Format yy.
. . .
    Move Year-Of-Birth-Exp to Year-Of-Birth-Win.
```

Si `Year-Of-Birth-Exp` contiene '1925', `Year-Of-Birth-Win` contendrá '25'. Sin embargo, si la ventana del siglo es 1930-2029, las referencias posteriores a `Year-Of-Birth-Win` lo tratarán como 2025, lo cual es incorrecto.

Parte 7. Mejora del rendimiento y la productividad

Capítulo 27. Ajuste del programa

Cuando un programa es comprensible, puede evaluar su rendimiento. Un flujo de control enredado hace que un programa sea difícil de entender y mantener, e inhibe la optimización de su código.

Acerca de esta tarea

Para mejorar el rendimiento del programa, examine al menos estos aspectos:

- Algoritmos subyacentes: Para un mejor rendimiento, el uso de algoritmos de sonido es esencial. Por ejemplo:
  - Un algoritmo sofisticado para clasificar un millón de elementos puede ser cientos de miles de veces más rápido que un algoritmo simple.
  - Si el programa accede con frecuencia a los datos, reduzca el número de pasos para acceder a los datos.
- Estructuras de datos: el uso de estructuras de datos adecuadas para los algoritmos es esencial.

Puede escribir programas que generen secuencias de código mejor generadas y utilicen los servicios del sistema de forma más eficiente. Estos aspectos adicionales pueden afectar al rendimiento:

- Técnicas de codificación: utilice un estilo de programación que permita al optimizador elegir tipos de datos eficientes y manejar las tablas de forma eficiente.
- Optimización: puede optimizar el código utilizando la opción de compilador `OPTIMIZE` .
- Opciones de compilador y `USE FOR DEBUGGING ON ALL PROCEDURES`: algunas opciones de compilador y lenguaje afectan a la eficiencia del programa.
- Entorno de ejecución: Considere la posibilidad de elegir opciones de tiempo de ejecución.
- **CICS**: para mejorar el tiempo de respuesta de la transacción, convierta las instancias de sentencias `EXEC CICS LINK` en sentencias `CALL` .

  Para obtener información sobre cómo mejorar el rendimiento de las llamadas dinámicas en CICS, consulte las tareas relacionadas.

Conceptos relacionados
“Optimización” en la página 533

Tareas relacionadas
“Ajuste del rendimiento de llamadas dinámicas en CICS” en la página 412
“Utilización de un estilo de programación óptimo” en la página 525
“Elección de tipos de datos eficientes” en la página 528
“Manejo eficiente de tablas” en la página 529
“Optimización del código” en la página 533
“Elección de características de compilador para mejorar el rendimiento” en la página 533
“Mejora del rendimiento de SFS” en la página 159

Referencias relacionadas
Capítulo 15, “Opciones de tiempo de ejecución”, en la página 315
“Opciones de compilador relacionadas con el rendimiento” en la página 534

Utilización de un estilo de programación óptimo

El estilo de codificación que utilice puede afectar al modo en que el optimizador maneja el código. Puede mejorar la optimización utilizando técnicas de programación estructurada, expresiones de factorización, utilizando constantes simbólicas y agrupando cálculos constantes y duplicados.

Acerca de esta tarea

Tareas relacionadas
“Utilización de programación estructurada” en la página 526
“Factorización de expresiones” en la página 526
“Utilización de constantes simbólicas” en la página 526
“Agrupación de cálculos constantes” en la página 527
“Agrupación de cálculos duplicados” en la página 527

Utilización de programación estructurada

El uso de sentencias de programación estructuradas, como EVALUATE y en línea PERFORM, hace que el programa sea más comprensible y genera un flujo de control más lineal. Como resultado, el optimizador puede operar sobre regiones más grandes del programa, lo que le proporciona un código más eficiente.

Acerca de esta tarea

Utilice construcciones de programación de arriba a abajo. Las sentencias PERFORM fuera de línea son un medio natural de realizar una programación descendente. Las sentencias PERFORM fuera de línea a menudo pueden ser tan eficientes como las sentencias PERFORM en línea, porque el optimizador puede simplificar o eliminar el código de enlace.

Evite utilizar las construcciones siguientes:

- Sentencias ALTER
- Ramas hacia atrás (excepto si es necesario para bucles para los que PERFORM no es adecuado).
- Procedimientos de PERFORM que implican un flujo de control irregular (por ejemplo, impedir que el control pase al final del procedimiento y vuelva a la sentencia PERFORM)

Factorización de expresiones

Al factorizar expresiones en sus programas, potencialmente puede eliminar una gran cantidad de cálculo innecesario.

Acerca de esta tarea

Por ejemplo, el primer bloque de código siguiente es más eficiente que el segundo bloque de código:

```
MOVE ZERO TO TOTAL
PERFORM VARYING I FROM 1 BY 1 UNTIL I = 10
  COMPUTE TOTAL = TOTAL + ITEM(I)
END-PERFORM
COMPUTE TOTAL = TOTAL * DISCOUNT
```

```
MOVE ZERO TO TOTAL
PERFORM VARYING I FROM 1 BY 1 UNTIL I = 10
  COMPUTE TOTAL = TOTAL + ITEM(I) * DISCOUNT
END-PERFORM
```

El optimizador no factoriza expresiones.

Utilización de constantes simbólicas

Para que el optimizador reconozca un elemento de datos como una constante en todo el programa, inicialícelo con una cláusula VALUE y no lo cambie en ningún lugar del programa.

Acerca de esta tarea

Si pasa un elemento de datos a un subprograma BY REFERENCE, el optimizador lo trata como un elemento de datos externo y presupone que se cambia en cada llamada de subprograma.

Si mueve un literal a un elemento de datos, el optimizador reconoce el elemento de datos como una constante sólo en un área limitada del programa después de la sentencia MOVE .

Agrupación de cálculos constantes

Cuando varios elementos de una expresión son constantes, asegúrese de que el optimizador puede optimizarlos. El compilador está enlazado por las reglas de evaluación de izquierda a derecha de COBOL. Por lo tanto, mueva todas las constantes al lado izquierdo de la expresión o agruparlas entre paréntesis.

Acerca de esta tarea

Por ejemplo, si V1, V2y V3 son variables y C1, C2y C3 son constantes, las expresiones de la izquierda a continuación son preferibles a las expresiones correspondientes de la derecha:

Más eficiente	**Menos eficiente**
`V1 * V2 * V3 * (C1 * C2 * C3)`	`V1 * V2 * V3 * C1 * C2 * C3`
`C1 + C2 + C3 + V1 + V2 + V3`	`V1 + C1 + V2 + C2 + V3 + C3`

En la programación de producción, a menudo hay una tendencia a colocar factores constantes en el lado derecho de las expresiones. Sin embargo, dicha colocación puede dar como resultado un código menos eficiente porque se pierde la optimización.

Agrupación de cálculos duplicados

Cuando los componentes de distintas expresiones estén duplicados, asegúrese de que el compilador pueda optimizarlos. Para expresiones aritméticas, el compilador está enlazado por las reglas de evaluación de izquierda a derecha de COBOL. Por lo tanto, mueva todos los duplicados al lado izquierdo de las expresiones o agruparlos entre paréntesis.

Acerca de esta tarea

Si V1 a V5 son variables, el cálculo V2 * V3 * V4 es un duplicado (conocido como una subexpresión común) en las dos sentencias siguientes:

```
COMPUTE A = V1 * (V2 * V3 * V4)
COMPUTE B = V2 * V3 * V4 * V5
```

En el ejemplo siguiente, V2 + V3 es una subexpresión común:

```
COMPUTE C = V1 + (V2 + V3)
COMPUTE D = V2 + V3 + V4
```

En el ejemplo siguiente, no hay ninguna subexpresión común:

```
COMPUTE A = V1 * V2 * V3 * V4
COMPUTE B = V2 * V3 * V4 * V5
COMPUTE C = V1 + (V2 + V3)
COMPUTE D = V4 + V2 + V3
```

El optimizador puede eliminar cálculos duplicados. No es necesario introducir cálculos temporales artificiales; un programa es a menudo más comprensible y más rápido sin ellos.

Elección de tipos de datos eficientes

Elegir el tipo de datos adecuado y la cláusula PICTURE puede producir un código más eficiente, al igual que evitar elementos de datos USAGE DISPLAY y USAGE NATIONAL en áreas que se utilizan mucho para los cálculos.

Acerca de esta tarea

Hacer un elemento de datos demasiado grande puede reducir el rendimiento, pero un elemento de datos cuya longitud es una potencia pequeña de 2 bytes (1, 2, 4, 8 o 16), y que está alineado en una potencia de límite de 2 bytes que coincide con su tamaño normalmente se puede inicializar y mover más rápidamente y con menos instrucciones que uno con una longitud o alineación impar.

La aritmética es más rápida con binario que con decimal empaquetado, que es más rápida que decimal con zona o DISPLAY, que es más rápida que decimal nacional.

Las opciones que afectan a los tipos también pueden afectar al rendimiento. Por ejemplo, FLOAT(BE) es más caro que FLOAT(NATIVE).

Los tipos de datos coherentes pueden reducir la necesidad de conversiones durante las operaciones en elementos de datos. También puede mejorar el rendimiento del programa determinando cuidadosamente cuándo utilizar los tipos de datos de coma fija y coma flotante.

Conceptos relacionados
"Formatos para datos numéricos" en la página 39

Tareas relacionadas
"Elección de elementos de datos computacionales eficientes" en la página 528
"Utilización de tipos de datos coherentes" en la página 529
"Cómo hacer que las expresiones aritméticas sean eficientes" en la página 529
"Hacer que las exponencias sean eficientes" en la página 529

Elección de elementos de datos computacionales eficientes

Cuando utilice un elemento de datos principalmente para la aritmética o como subíndice, codifique USAGE BINARY en la entrada de descripción de datos para el elemento. Las operaciones para manipular datos binarios son más rápidas que las operaciones para manipular datos decimales.

Acerca de esta tarea

Sin embargo, si una sentencia aritmética de coma fija tiene resultados intermedios con una precisión grande (número de dígitos significativos), el compilador utiliza la aritmética decimal de todos modos, después de convertir los operandos a formato decimal empaquetado. Para sentencias aritméticas de punto fijo, el compilador normalmente utiliza aritmética binaria para cálculos simples con operandos binarios si la precisión es de ocho o menos dígitos. Por encima de 18 dígitos, el compilador siempre utiliza la aritmética decimal. Con una precisión de nueve a 18 dígitos, el compilador utiliza cualquiera de los dos formatos.

Para producir el código más eficiente para un elemento de datos BINARY , asegúrese de que tiene:

- Un signo (un S en su cláusula PICTURE)
- Ocho o menos dígitos

Para un elemento de datos que tenga más de ocho dígitos o se utilice con elementos de datos DISPLAY o NATIONAL , utilice PACKED-DECIMAL.

Para producir el código más eficiente para un elemento de datos PACKED-DECIMAL , asegúrese de que tiene:

- Un signo (un S en su cláusula PICTURE)
- Un número impar de dígitos (9s en la cláusula PICTURE), de modo que ocupa un número exacto de bytes sin un medio byte sobrante

Utilización de tipos de datos coherentes

En operaciones en operandos de tipos diferentes, uno de los operandos debe convertirse al mismo tipo que el otro. Cada conversión requiere varias instrucciones. Por ejemplo, es posible que sea necesario escalar uno de los operandos para proporcionarle el número adecuado de posiciones decimales.

Acerca de esta tarea

Puede evitar en gran medida las conversiones utilizando tipos de datos coherentes y proporcionando a ambos operandos el mismo uso y también las especificaciones PICTURE adecuadas. Es decir, debe asegurarse de que dos números que se van a comparar, añadir o restar no sólo tienen el mismo uso, sino también el mismo número de posiciones decimales (9s después de V en la cláusula PICTURE).

Cómo hacer que las expresiones aritméticas sean eficientes

El cálculo de expresiones aritméticas que se evalúan en coma flotante es más eficiente cuando los operandos necesitan poca o ninguna conversión. Utilice operandos que sean COMP-1 o COMP-2 para producir el código más eficiente.

Acerca de esta tarea

Defina elementos enteros como BINARY o PACKED-DECIMAL con nueve o menos dígitos para permitir una conversión rápida a datos de coma flotante. Además, la conversión de un elemento COMP-1 o COMP-2 a un entero de punto fijo con nueve o menos dígitos, sin SIZE ERROR en vigor, es eficaz cuando el valor del elemento COMP-1 o COMP-2 es inferior a 1.000.000.000.

Hacer que las exponencias sean eficientes

Utilice la coma flotante para exponencias para grandes exponentes para lograr una evaluación más rápida y resultados más precisos.

Acerca de esta tarea

Por ejemplo, la primera sentencia siguiente se calcula de forma más rápida y precisa que la segunda sentencia:

```
COMPUTE fixed-point1 = fixed-point2 ** 100000.E+00
COMPUTE fixed-point1 = fixed-point2 ** 100000
```

Un exponente de coma flotante hace que se utilice la aritmética de coma flotante para calcular la exponenciación.

Manejo eficiente de tablas

Puede utilizar varias técnicas para mejorar la eficiencia de las operaciones de manejo de tablas y para influir en el optimizador. La devolución de sus esfuerzos puede ser significativa, especialmente cuando las operaciones de manejo de tablas son una parte importante de una aplicación.

Acerca de esta tarea

Las dos directrices siguientes afectan a la elección de cómo hacer referencia a los elementos de tabla:

- Utilice la indexación en lugar de la suscripción.

  Aunque el compilador puede eliminar índices y subíndices duplicados, la referencia original a un elemento de tabla es más eficiente con los índices (incluso si los subíndices eran BINARY). El valor de un índice tiene el tamaño de elemento factorizado en él, mientras que el valor de un subíndice debe multiplicarse por el tamaño de elemento cuando se utiliza el subíndice. El índice ya contiene el

desplazamiento desde el inicio de la tabla, y este valor no tiene que calcularse en tiempo de ejecución. Sin embargo, la suscripción puede ser más fácil de entender y mantener.

- Utilice la indexación relativa.

  Las referencias de índice relativas (es decir, las referencias en las que se añade o se resta un literal numérico sin signo del nombre-índice) se ejecutan al menos tan rápido como las referencias de índice directas, y a veces más rápido. No hay ningún mérito en mantener los índices alternativos con la compensación factorizada.

Tanto si utiliza índices como subíndices, las siguientes directrices de codificación pueden ayudarle a obtener un mejor rendimiento:

- Coloque índices o subíndices constantes y duplicados a la izquierda.

  Puede reducir o eliminar los cálculos de tiempo de ejecución de esta forma. Incluso cuando todos los índices o subíndices son variables, intente utilizar las tablas para que el subíndice situado más a la derecha varíe con más frecuencia para las referencias que se producen cerca unas de otras en el programa. Esta práctica también mejora el patrón de referencias de almacenamiento y también la paginación. Si todos los índices o subíndices son duplicados, el cálculo completo del índice o subíndice es una subexpresión común.

- Especifique la longitud del elemento para que coincida con la de las tablas relacionadas.

  Al indexar o subindexar tablas, es más eficaz si todas las tablas tienen la misma longitud de elemento. De esta forma, el paso para la última dimensión de las tablas es el mismo, y el optimizador puede reutilizar el índice o subíndice situado más a la derecha calculado para una tabla. Si tanto las longitudes de elemento como el número de apariciones en cada dimensión son iguales, las zanjas para las dimensiones distintas de la última también son iguales, lo que da como resultado una mayor concordancia entre sus cálculos de subíndice. A continuación, el optimizador puede reutilizar índices o subíndices que no sean los más a la derecha.

- Evite errores en las referencias codificando las comprobaciones de índice y subíndice en el programa.

  Si necesita validar índices y subíndices, puede ser más rápido codificar sus propias comprobaciones que utilizar la opción de compilador SSRANGE .

También puede mejorar la eficiencia de las tablas utilizando estas directrices:

- Utilice elementos de datos binarios para todos los subíndices.

  Cuando utilice subíndices para direccionar una tabla, utilice un elemento de datos firmado de BINARY con ocho o menos dígitos. En algunos casos, el uso de cuatro o menos dígitos para el elemento de datos también puede mejorar el tiempo de proceso.

- Utilice elementos de datos binarios para elementos de tabla de longitud variable.

  Para tablas con elementos de longitud variable, puede mejorar el código para OCCURS DEPENDING ON (ODO). Para evitar conversiones innecesarias cada vez que se haga referencia a los elementos de longitud variable, especifique BINARY para objetos OCCURS . . . DEPENDING ON .

- Utilice elementos de datos de longitud fija siempre que sea posible.

  La copia de elementos de datos de longitud variable en un elemento de datos de longitud fija antes de un periodo de uso de alta frecuencia puede reducir parte de la sobrecarga asociada con el uso de elementos de datos de longitud variable.

- Organice las tablas según el tipo de método de búsqueda utilizado.

  Si la tabla se busca secuencialmente, ponga los valores de datos con mayor probabilidad de satisfacer los criterios de búsqueda al principio de la tabla. Si la tabla se busca utilizando un algoritmo de búsqueda binaria, coloque los valores de datos en la tabla ordenados alfabéticamente en el campo de clave de búsqueda.

Conceptos relacionados
“Optimización de referencias de tabla” en la página 531

Tareas relacionadas
“Cómo hacer referencia a un elemento de una tabla” en la página 62
“Elección de tipos de datos eficientes” en la página 528

Referencias relacionadas
“SRANGE” en la página 299

Optimización de referencias de tabla

El compilador COBOL optimiza las referencias de tabla de varias maneras.

Para la referencia de elemento de tabla `ELEMENT(S1 S2 S3)`, donde S1, S2y S3 son subíndices, el compilador evalúa la expresión siguiente:

```
comp_s1 * d1 + comp_s2 * d2 + comp_s3 * d3 + base_address
```

Aquí `comp_s1` es el valor de S1 después de la conversión a binario, `comp-s2` es el valor de S2 después de la conversión a binario, y así sucesivamente. Los pasos para cada dimensión son d1, d2y d3. El *paso* de una dimensión determinada es la distancia en bytes entre elementos de tabla cuyos números de aparición en esa dimensión difieren en 1 y cuyos otros números de aparición son iguales. Por ejemplo, la zancada d2 de la segunda dimensión del ejemplo anterior es la distancia en bytes entre `ELEMENT(S1 1 S3)` y `ELEMENT(S1 2 S3)`.

Los cálculos de índice son similares a los cálculos de subíndice, excepto que no es necesario realizar ninguna multiplicación. Los valores de índice tienen la zancada factorizada en ellos. Implican la carga de los índices en registros, y estas transferencias de datos se pueden optimizar, al igual que los términos de cálculo de subíndice individuales se optimizan.

Puesto que el compilador evalúa las expresiones de izquierda a derecha, el optimizador encuentra la mayor cantidad de oportunidades para eliminar cálculos cuando la constante o los subíndices duplicados son los más a la izquierda.

Optimización de elementos constantes y variables

Supongamos que `C1, C2,..` son elementos de datos constantes y que `V1, V2,...` son elementos de datos variables. A continuación, para la referencia de elemento de tabla `ELEMENT(V1 C1 C2)` el compilador sólo puede eliminar los términos individuales `comp_c1 * d2` y `comp_c2 * d3` como constante de la expresión:

```
comp_v1 * d1 + comp_c1 * d2 + comp_c2 * d3 + base_address
```

Sin embargo, para la referencia de elemento de tabla `ELEMENT(C1 C2 V1)` el compilador puede eliminar toda la subexpresión `comp_c1 * d1 + comp_c2 * d2` como constante de la expresión:

```
comp_c1 * d1 + comp_c2 * d2 + comp_v1 * d3 + base_address
```

En la referencia de elemento de tabla `ELEMENT(C1 C2 C3)`, todos los subíndices son constantes, por lo que no se realiza ningún cálculo de subíndice en tiempo de ejecución. La expresión es:

```
comp_c1 * d1 + comp_c2 * d2 + comp_c3 * d3 + base_address
```

Con el optimizador, esta referencia puede ser tan eficiente como una referencia a un elemento escalar (no de tabla).

Optimización de elementos duplicados

En el elemento de tabla hace referencia a `ELEMENT(V1 V3 V4)` y `ELEMENT(V2 V3 V4)` sólo los términos individuales `comp_v3 * d2` y `comp_v4 * d3` son subexpresiones comunes en las expresiones necesarias para hacer referencia a los elementos de tabla:

```
comp_v1 * d1 + comp_v3 * d2 + comp_v4 * d3 + base_address
comp_v2 * d1 + comp_v3 * d2 + comp_v4 * d3 + base_address
```

Sin embargo, para las dos referencias de elemento de tabla `ELEMENT(V1 V2 V3)` y `ELEMENT(V1 V2 V4)` toda la subexpresión `comp_v1 * d1 + comp_v2 * d2` es común entre las dos expresiones necesarias para hacer referencia a los elementos de tabla:

```
comp_v1 * d1 + comp_v2 * d2 + comp_v3 * d3 + base_address
comp_v1 * d1 + comp_v2 * d2 + comp_v4 * d3 + base_address
```

En las dos referencias `ELEMENT(V1 V2 V3)` y `ELEMENT(V1 V2 V3)`, las expresiones son las mismas:

```
comp_v1 * d1 + comp_v2 * d2 + comp_v3 * d3 + base_address
comp_v1 * d1 + comp_v2 * d2 + comp_v3 * d3 + base_address
```

Con el optimizador, la segunda (y cualquier referencia posterior) al mismo elemento puede ser tan eficiente como una referencia a un elemento escalar (no de tabla).

Optimización de elementos de longitud variable

Un elemento de grupo que contiene un elemento de datos `OCCURS DEPENDING ON` subordinado tiene una longitud variable. El programa debe realizar un código especial cada vez que se haga referencia a un elemento de datos de longitud variable.

Puesto que este código está fuera de línea, puede interrumpir la optimización. Además, el código para manipular elementos de datos de longitud variable es mucho menos eficiente que el de los elementos de datos de tamaño fijo y puede aumentar significativamente el tiempo de proceso. Por ejemplo, el código para comparar o mover un elemento de datos de longitud variable puede implicar llamar a una rutina de biblioteca y es mucho más lento que el mismo código para los elementos de datos de longitud fija.

Comparación de la indexación directa y relativa

Las referencias de índice relativas son tan rápidas o más rápidas que las referencias de índice directas.

La indexación directa en `ELEMENT (I5, J3, K2)` requiere este preproceso:

```
SET I5 TO I
SET I5 UP BY 5
SET J3 TO J
SET J3 DOWN BY 3
SET K2 TO K
SET K2 UP BY 2
```

Este proceso hace que la indexación directa sea menos eficiente que la indexación relativa en `ELEMENT (I + 5, J - 3, K + 2)`.

Conceptos relacionados
“Optimización” en la página 533

Tareas relacionadas
“Manejo eficiente de tablas” en la página 529

Optimización del código

Cuando el programa esté listo para la prueba final, especifique la opción de compilador OPTIMIZE para que el código probado y el código de producción sean idénticos. Tenga en cuenta que IBM recomienda que todos los usuarios utilicen OPT(FULL) para obtener el mejor rendimiento.

Acerca de esta tarea

Si ejecuta con frecuencia un programa sin recompilarlo durante el desarrollo, es posible que también desee utilizar OPTIMIZE. Sin embargo, si vuelve a compilar con frecuencia, la sobrecarga de OPTIMIZE podría superar sus ventajas a menos que esté utilizando la expansión de lenguaje ensamblador (opción de compiladorLIST) para ajustar el programa.

Para las pruebas de unidad de un programa, probablemente le resultará más fácil depurar código que no se ha optimizado.

Para ver cómo funciona el optimizador en un programa, compílelo con y sin utilizar OPTIMIZE y compare el código generado. (Utilice la opción de compilador LIST para solicitar el listado de ensamblador del código generado.)

Conceptos relacionados
"Optimización" en la página 533

Referencias relacionadas
"lista" en la página 287
"OPTIMIZAR" en la página 291

Optimización

Para mejorar la eficiencia del código generado, puede utilizar la opción de compilador OPTIMIZE .

OPTIMIZE hace que el optimizador COBOL realice las optimizaciones siguientes:

- Elimine las transferencias innecesarias de control y las ramificaciones ineficientes, incluidas las generadas por el compilador que no son evidentes al mirar el programa fuente.
- Siempre que sea posible, el optimizador coloca las sentencias en línea, eliminando la necesidad de código de enlace. Esta optimización se conoce como *integración de procedimientos*.
- Elimine los cálculos duplicados (como los cálculos de subíndice y las sentencias repetidas) que no tienen ningún efecto en los resultados del programa.
- Elimine los cálculos constantes realizándolos cuando se compila el programa.
- Eliminar expresiones condicionales constantes.
- Agregar movimientos de elementos contiguos (como los que a menudo se producen con el uso de MOVE CORRESPONDING) en un solo movimiento. Tanto el origen como el destino deben ser contiguos para que se agreguen los movimientos.
- Descarte los elementos de datos no referenciados de DATA DIVISIONy suprima la generación de código para inicializar estos elementos de datos en sus cláusulas VALUE . (El optimizador realiza esta acción sólo cuando utiliza la FULL subopción.)

Elección de características de compilador para mejorar el rendimiento

La elección de opciones de compilador relacionadas con el rendimiento y el uso de la sentencia USE FOR DEBUGGING ON ALL PROCEDURES pueden afectar a la optimización del programa.

Acerca de esta tarea

Es posible que tenga un sistema personalizado que requiera determinadas opciones para obtener un rendimiento óptimo. Siga estos pasos:

Procedimiento

1. Para ver cuáles son los valores predeterminados del sistema, obtenga un listado corto para cualquier programa y revise los valores de opción listados.
2. Seleccionar opciones relacionadas con el rendimiento para compilar los programas.

   Importante: Consultar con el programador del sistema cómo ajustar los programas COBOL. Al hacerlo, se asegurará de que las opciones que elija sean adecuadas para los programas de su sitio.

Resultados

Otra característica del compilador a tener en cuenta es la sentencia `USE FOR DEBUGGING ON ALL PROCEDURES` . Puede afectar en gran medida al optimizador del compilador. La opción `ON ALL PROCEDURES` genera código adicional en cada transferencia a un nombre de procedimiento. Aunque es muy útil para la depuración, puede hacer que el programa sea significativamente más grande e inhibir la optimización sustancialmente.

Conceptos relacionados
"Optimización" en la página 533

Tareas relacionadas
"Optimización del código" en la página 533
"Obtención de listados" en la página 382

Referencias relacionadas
"Opciones de compilador relacionadas con el rendimiento" en la página 534

Opciones de compilador relacionadas con el rendimiento

En la tabla siguiente puede ver una descripción de la finalidad de cada opción, sus ventajas y desventajas de rendimiento y las notas de uso cuando sea aplicable.

Tabla 47. ***Opciones de compilador relacionadas con el rendimiento***

Opción de compilador	objetivo	Ventajas de rendimiento	Desventajas de rendimiento	Notas de uso
`ARITH(EXTEND)` (consulte "HARIT" en la página 270)	Para aumentar el número máximo de dígitos permitidos para números decimales	Ninguna	`ARITH(EXTEND)` provoca cierta degradación en el rendimiento para todos los tipos de datos decimales debido a resultados intermedios más grandes.	La cantidad de degradación que experimenta depende directamente de la cantidad de datos decimales que utilice.
`DYNAM` (consulte "DYNAM" en la página 281)	Para que los subprogramas (llamados a través de la sentencia `CALL`) se carguen dinámicamente en tiempo de ejecución	Los subprogramas son más fáciles de mantener, porque no es necesario volver a editar los enlaces si se cambia un subprograma.	Hay una ligera penalización en el rendimiento, porque la llamada debe pasar por una rutina biblioteca .	Para liberar almacenamiento virtual que ya no es necesario, emita la sentencia `CANCEL` .
`OPTIMIZE(STD)` (consulte "OPTIMIZAR" en la página 291)	Para optimizar el código generado para un mejor rendimiento	Generalmente da como resultado un código de tiempo de ejecución más eficiente	Tiempo de compilación más largo: `OPTIMIZE` requiere más tiempo de proceso para compilaciones que `NOOPTIMIZE`.	`NOOPTIMIZE` se utiliza generalmente durante el desarrollo del programa cuando se necesitan compilaciones frecuentes; también permite la depuración simbólica. Para las ejecuciones de producción, se recomienda `OPTIMIZE` .

*Tabla 47. **Opciones de compilador relacionadas con el rendimiento** (continuación)*

Opción de compilador	objetivo	Ventajas de rendimiento	Desventajas de rendimiento	Notas de uso
`OPTIMIZE(FULL)`	Para optimizar el código generado para un mejor rendimiento y también optimizar el `DATA DIVISION`	Generalmente da como resultado un código de tiempo de ejecución más eficiente y un menor uso de almacenamiento	Tiempo de compilación más largo: `OPTIMIZE` requiere más tiempo de proceso para compilaciones que `NOOPTIMIZE`.	`OPT(FULL)` suprime elementos de datos no utilizados, lo que puede no ser deseable en el caso de indicaciones de fecha y hora o elementos de datos que sólo se utilizan como marcadores para la lectura de volcado.
`SSRANGE` (consulte "SRANGE" en la página 299)	Para verificar que todas las referencias de tabla y expresiones de modificación de referencia están en límites adecuados	`SSRANGE` genera código adicional para verificar referencias de tabla. El uso de `NOSSRANGE` hace que no se genere ese código.	Con `SSRANGE` especificado, las comprobaciones de rangos válidos afectan al rendimiento del compilador.	En general, si necesita verificar las referencias de tabla sólo unas pocas veces en lugar de hacerlo en cada referencia, la codificación de sus propias comprobaciones puede ser más rápida que la utilización de `SSRANGE`. Puede desactivar `SSRANGE` en tiempo de ejecución utilizando la opción de tiempo de ejecución `CHECK(OFF)` . Para aplicaciones sensibles al rendimiento, se recomienda `NOSSRANGE` .
`NOTEST` (consulte "TEST" en la página 301)	Para evitar el código de objeto adicional que es necesario para aprovechar al máximo el depurador.	Ninguna	`TEST` amplía significativamente el archivo de objeto porque añade información de depuración. Al enlazar el programa, puede indicar al enlazador que excluya la información de depuración, lo que da como resultado aproximadamente el mismo tamaño ejecutable que se crearía si los módulos se compilaran con `NOTEST`. Si se incluye la información de depuración, puede producirse una ligera degradación del rendimiento porque un ejecutable más grande tarda más en cargarse y podría aumentar la paginación.	`TESTNOOPT`Para ejecuciones de producción, se recomienda utilizar `NOTEST` .
`TRUNC(OPT)` (consulte "TRUNC" en la página 302)	Para evitar que se genere código para truncar los campos de recepción de las operaciones aritméticas	No genera código adicional y, por lo general, mejora el rendimiento	Tanto `TRUNC(BIN)` como `TRUNC(STD)` generan código adicional siempre que se cambia un elemento de datos `BINARY` . `TRUNC(BIN)` suele ser la más lenta de estas opciones.	`TRUNC(STD)` se ajusta a 85 COBOL Estándar, pero `TRUNC(BIN)` y `TRUNC(OPT)` no. Con `TRUNC(OPT)`, el compilador presupone que los datos se ajustan a las especificaciones `PICTURE` y `USAGE` . Se recomienda `TRUNC(OPT)` siempre que sea posible.

Conceptos relacionados

"Optimización" en la página 533

Tareas relacionadas

"Generación de una lista de mensajes del compilador" en la página 246
"Elección de características de compilador para mejorar el rendimiento" en la página 533
"Manejo eficiente de tablas" en la página 529

Referencias relacionadas
“Representación de signo de datos con zona y decimal empaquetado” en la página 47
“opciones de compilador” en la página 265

Evaluación del rendimiento

Rellene la siguiente hoja de trabajo para ayudarle a evaluar el rendimiento del programa. Si responde afirmativamente a cada pregunta, es probable que esté mejorando el rendimiento.

Acerca de esta tarea

Al pensar en la compensación de rendimiento, asegúrese de comprender la función de cada opción, así como las ventajas y desventajas de rendimiento. Es posible que prefiera la función en lugar de aumentar el rendimiento en muchas instancias.

Tabla 48. ***Hoja de trabajo de ajuste de rendimiento***

Opción de compilador	**Consideraciones**	**¿Sí?**
DYNAM	¿Puede utilizar NODYNAM? Considere las compensaciones de rendimiento.	
OPTIMIZE	¿Se utiliza OPTIMIZE para las ejecuciones de producción? ¿Puede utilizar OPTIMIZE(FULL)?	
SSRANGE	¿Utiliza NOSSRANGE para las ejecuciones de producción?	
TEST	¿Utiliza NOTEST para ejecuciones de producción?	
TRUNC	¿Utiliza TRUNC(OPT) cuando sea posible?	

Tareas relacionadas
“Elección de características de compilador para mejorar el rendimiento” en la página 533

Referencias relacionadas
“Opciones de compilador relacionadas con el rendimiento” en la página 534

Capítulo 28. Simplificación de la codificación

Acerca de esta tarea

Puede utilizar técnicas de codificación para mejorar su productividad. Utilizando la sentencia COPY , COBOL y los servicios invocables de , puede evitar la codificación repetitiva y tener que codificar muchos cálculos aritméticos u otras tareas complejas.

Si el programa contiene secuencias de código utilizadas con frecuencia (como bloques de elementos de datos comunes, rutinas de entrada-salida, rutinas de error o incluso programas COBOL completos), escriba las secuencias de código una vez y póngarlas en una biblioteca de copia COBOL. Puede utilizar la sentencia COPY para recuperar estas secuencias de código y que se incluyan en el programa durante la compilación. El uso de libros de copias de esta manera elimina la codificación repetitiva.

COBOL proporciona diversas prestaciones para manipular series y números. Estas prestaciones pueden ayudarle a simplificar la codificación.

Los servicios invocables de fecha y hora de almacenan las fechas como enteros binarios de palabra completa y almacenan las indicaciones de fecha y hora como valores de coma flotante largos (64 bits). Estos formatos le permiten realizar cálculos aritméticos sobre los valores de fecha y hora de forma sencilla y eficiente. No es necesario escribir subrutinas especiales que utilicen servicios fuera de la biblioteca de lenguaje para realizar dichos cálculos.

Tareas relacionadas
“Utilización de funciones intrínsecas numéricas” en la página 50
“Eliminación de codificación repetitiva” en la página 537
“Conversión de elementos de datos (funciones intrínsecas)” en la página 106
“Evaluación de elementos de datos (funciones intrínsecas)” en la página 109
“Manipulación de fechas y horas” en la página 539

Eliminación de codificación repetitiva

Para incluir sentencias fuente almacenadas en un programa, utilice la sentencia COPY en cualquier división de programa y en cualquier nivel de secuencia de código. Puede anidar sentencias COPY a cualquier profundidad.

Acerca de esta tarea

Para especificar más de una biblioteca de copia, establezca la variable de entorno SYSLIB en varios nombres de vía de acceso separados por un dos puntos (:), o defina sus propias variables de entorno e incluya la frase siguiente en la sentencia COPY :

```
IN/OF library-name
```

Por ejemplo:

```
COPY MEMBER1 OF COPYLIB
```

Si omite esta frase calificada, el valor predeterminado es SYSLIB.

COPY y línea de depuración: Para que el texto copiado se trate como líneas de depuración, por ejemplo, como si hubiera un D insertado en la columna 7, coloque D en la primera línea de la sentencia COPY . Una sentencia COPY no puede ser una línea de depuración; si contiene un Dy no se especifica la modalidad WITH DEBUGGING , no obstante, se procesa la sentencia COPY .

“Ejemplo: utilización de la sentencia COPY” en la página 538

Referencias relacionadas
Capítulo 14, “Sentencias de direccionamiento de compilador”, en la página 309

Ejemplo: utilización de la sentencia COPY

Estos ejemplos muestran cómo puede utilizar la sentencia COPY para incluir texto de biblioteca en un programa.

Supongamos que la entrada de biblioteca CFILEA consta de las siguientes entradas FD :

```
    BLOCK CONTAINS 20 RECORDS
    RECORD CONTAINS 120 CHARACTERS
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS FILE-OUT.
01  FILE-OUT       PIC X(120).
```

Puede recuperar el nombre de texto CFILEA utilizando la sentencia COPY en un programa fuente como se indica a continuación:

```
FD FILEA
         COPY CFILEA.
```

La entrada de biblioteca se copia en el programa y el listado de programas resultante tiene el siguiente aspecto:

```
FD FILEA
         COPY CFILEA.
C    BLOCK CONTAINS 20 RECORDS
C    RECORD CONTAINS 120 CHARACTERS
C    LABEL RECORDS ARE STANDARD
C    DATA RECORD IS FILE-OUT.
C    01  FILE-OUT   PIC X(120).
```

En el listado fuente del compilador, la sentencia COPY se imprime en una línea aparte. C precede a las líneas copiadas.

Supongamos que un libro de copias con el nombre de texto DOWORK se almacena utilizando las sentencias siguientes:

```
COMPUTE QTY-ON-HAND = TOTAL-USED-NUMBER-ON-HAND
MOVE QTY-ON-HAND to PRINT-AREA
```

Para recuperar el libro de copias identificado como DOWORK, codifique:

```
paragraph-name.
    COPY DOWORK.
```

Las sentencias que se encuentran en el procedimiento DOWORK seguirán a *nombre-párrafo*.

Si utiliza la opción de compilador EXIT para proporcionar un módulo LIBEXIT , los resultados pueden diferir de los que se muestran aquí.

Tareas relacionadas
“Eliminación de codificación repetitiva” en la página 537

Referencias relacionadas
Capítulo 14, “Sentencias de direccionamiento de compilador”, en la página 309

Manipulación de fechas y horas

Para invocar un servicio invocable de fecha u hora, utilice una sentencia CALL con los parámetros correctos para ese servicio. Defina los elementos de datos para la sentencia CALL en DATA DIVISION con las definiciones de datos que necesita ese servicio.

Acerca de esta tarea

```
77  argument          pic  s9(9)  comp.
01  format.
    05 format-length  pic  s9(4)  comp.
    05 format-string  pic  x(80).
77  result            pic  x(80).
77  feedback-code     pic  x(12)  display.
. . .
    CALL "CEEDATE" using argument, format, result, feedback-code.
```

En el ejemplo anterior, el servicio invocable CEEDATE convierte un número que representa una fecha Lilian en el elemento de datos `argument` a una fecha en formato de caracteres, que se graba en el elemento de datos `result`. La serie de imagen contenida en el elemento de datos `format` controla el formato de la conversión. La información sobre el éxito o el error de la llamada se devuelve en el elemento de datos `feedback-code`.

En las sentencias CALL que utilice para invocar los servicios invocables de fecha y hora, debe utilizar un literal para el nombre de programa en lugar de un identificador.

Un programa llama a los servicios invocables de fecha y hora utilizando el convenio de enlace del sistema estándar.

“Ejemplo: manipulación de fechas” en la página 540

Conceptos relacionados
Apéndice C, “Servicios invocables de fecha y hora”, en la página 561

Tareas relacionadas
“Obtención de comentarios de los servicios invocables de fecha y hora” en la página 539
“Manejo de condiciones de servicios invocables de fecha y hora” en la página 540

Referencias relacionadas
“Señal de comentarios” en la página 541
“Términos y series de caracteres de imagen” en la página 542
sentencia CALL (*COBOL for Linux en x86 Consulta de lenguaje*)

Obtención de comentarios de los servicios invocables de fecha y hora

Puede especificar un parámetro de código de comentarios (que es opcional) en cualquier servicio invocable de fecha y hora. Especifique OMITTED para este parámetro si no desea que el servicio devuelva información sobre el éxito o el error de la llamada.

Acerca de esta tarea

Sin embargo, si no especifica este parámetro y el servicio invocable no se completa satisfactoriamente, el programa terminará de forma anómala.

Cuando llama a un servicio invocable de fecha y hora y especifica OMITTED para el código de comentarios, el registro especial RETURN-CODE se establece en 0 si el servicio es satisfactorio, pero no se modifica si el servicio no es satisfactorio. Si el código de comentarios no es OMITTED, el registro especial RETURN-CODE siempre se establece en 0 independientemente de si el servicio se ha completado correctamente.

“Ejemplo: formato de fechas para salida” en la página 540

Referencias relacionadas
“Señal de comentarios” en la página 541

Manejo de condiciones de servicios invocables de fecha y hora

El manejo de condiciones por parte de COBOL para Linux es significativamente diferente del proporcionado por IBM Language Environment en el host. COBOL para Linux se adhiere al esquema de manejo de condiciones COBOL nativo y no proporciona el nivel de soporte que se encuentra en Entorno de lenguajes.

Acerca de esta tarea

Si pasa una señal de retroalimentación un argumento, simplemente se devolverá después de que se haya rellenado la información adecuada. Puede codificar la lógica en la rutina de llamada para examinar el contenido y realizar cualquier acción si es necesario. La condición no se señalará.

Referencias relacionadas
“Señal de comentarios” en la página 541

Ejemplo: manipulación de fechas

El ejemplo siguiente muestra cómo utilizar los servicios invocables de Fecha y hora para convertir una fecha a un formato diferente y realizar un cálculo simple con la fecha formateada.

```
CALL CEEDAYS USING dateof_hire, 'YYMMDD', doh_lilian, fc.
CALL CEELOCT USING todayLilian, today_seconds, today_Gregorian, fc.
COMPUTE servicedays  = today_Lilian - doh_Lilian.
COMPUTE serviceyears = service_days / 365.25.
```

El ejemplo anterior utiliza la fecha original de contratación en el formato AAMMDD para calcular el número de años de servicio para un empleado. El cálculo es el siguiente:

1. Llame a CEEDAYS (Convertir fecha a formato Lilian) para convertir la fecha a formato Lilian.
2. Llame a CEELOCT (Obtener hora local actual) para obtener la hora local actual.
3. Reste doh_Lilian de today_Lilian (el número de días desde el principio del calendario gregoriano hasta la hora local actual) para calcular el número de días de empleo del empleado.
4. Divida el número de días por 365,25 para obtener el número de años de servicio.

Ejemplo: formato de fechas para salida

El ejemplo siguiente utiliza los servicios invocables de Fecha y hora para formatear y visualizar una fecha obtenida de una sentencia ACCEPT .

Muchos servicios invocables ofrecen prestaciones que de otro modo requerirían una codificación extensa. Dos de estos servicios son CEEDAYS y CEEDATE, que puede utilizar de forma eficaz cuando desee formatear fechas.

```
CBL QUOTE
       ID DIVISION.
       PROGRAM-ID. HOHOHO.
      *************************************************************
      * FUNCTION:  DISPLAY TODAY'S DATE IN THE FOLLOWING FORMAT: *
      *            WWWWWWWWW, MMMMMMMM DD, YYYY                  *
      *                                                          *
      *            For example:  MONDAY, OCTOBER 18, 2010        *
      *                                                          *
      *************************************************************
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.

       01   CHRDATE.
            05 CHRDATE-LENGTH     PIC S9(4) COMP VALUE 10.
```

```
           05 CHRDATE-STRING     PIC X(10).
       01   PICSTR.
           05 PICSTR-LENGTH      PIC  S9(4) COMP.
           05 PICSTR-STRING      PIC  X(80).

       77   LILIAN PIC           S9(9) COMP.
       77   FORMATTED-DATE       PIC X(80).

       PROCEDURE DIVISION.
      ***************************************************************
      *    USE  DATE/TIME CALLABLE SERVICES TO PRINT OUT            *
      *    TODAY'S DATE FROM COBOL ACCEPT STATEMENT.                *
      ***************************************************************
           ACCEPT CHRDATE-STRING FROM DATE.

           MOVE "YYMMDD" TO PICSTR-STRING.
           MOVE 6 TO PICSTR-LENGTH.
           CALL "CEEDAYS" USING CHRDATE , PICSTR , LILIAN , OMITTED.

           MOVE " WWWWWWWWWZ, MMMMMMMMMZ DD, YYYY " TO PICSTR-STRING.
           MOVE 50 TO PICSTR-LENGTH.
           CALL "CEEDATE" USING LILIAN , PICSTR , FORMATTED-DATE ,
                 OMITTED.

           DISPLAY "******************************".
           DISPLAY FORMATTED-DATE.
           DISPLAY "******************************".

           STOP RUN.
```

Señal de comentarios

Una señal de comentarios contiene información de comentarios en forma de una señal de condición. La señal de condición establecida por el servicio invocable se devuelve a la rutina de llamada, indicando si el servicio se ha completado correctamente.

COBOL para Linux utiliza la misma señal de comentarios que Entorno de lenguajes, que se define de la forma siguiente:

```
01  FC.
    02  Condition-Token-Value.
    COPY  CEEIGZCT.
        03  Case-1-Condition-ID.
            04  Severity    PIC S9(4) COMP.
            04  Msg-No      PIC S9(4) COMP.
        03  Case-2-Condition-ID
                  REDEFINES Case-1-Condition-ID.
            04  Class-Code  PIC S9(4) COMP.
            04  Cause-Code  PIC S9(4) COMP.
        03  Case-Sev-Ctl    PIC X.
        03  Facility-ID     PIC XXX.
    02  I-S-Info            PIC S9(9) COMP.
```

El contenido de cada campo y las diferencias de IBM Language Environment en el host son las siguientes:

Severity
Este es el número de gravedad con los siguientes valores posibles:

0
Sólo información (o, si toda la señal es cero, sin información)

1
Aviso: servicio completado, probablemente correctamente

2
Se ha detectado un error: se ha intentado la corrección; el servicio se ha completado, quizás incorrectamente

3
Error grave: el servicio no se ha completado

4
Error crítico: el servicio no se ha completado

Msg-No
Es el número de mensaje asociado.

Case-Sev-Ctl
Este campo siempre contiene el valor 1.

Facility-ID
Este campo siempre contiene los caracteres CEE.

I-S-Info
Este campo siempre contiene el valor 0.

El libro de copias de ejemplo CEEIGZCT.CPY define las señales de condición. Los símbolos de condición del archivo son equivalentes a los proporcionados por Entorno de lenguajes, excepto que las representaciones de caracteres están en ASCII en lugar de EBCDIC. Debe tener en cuenta estas diferencias si compara las señales de condición con las proporcionadas por Entorno de lenguajes.

Las descripciones de los servicios invocables individuales incluyen una lista de los códigos de comentarios simbólicos que se pueden devolver en el campo de salida de código de comentarios especificado en la invocación del servicio. Además de estos, es posible que se devuelva el código de retorno simbólico CEE0PD para cualquier servicio invocable. Consulte el mensaje IWZ0813S para obtener más detalles.

Todos los servicios invocables de Fecha y hora se basan en el calendario gregoriano. Las variables de fecha asociadas a este calendario tienen límites de arquitectura. Estos límites son:

Fecha de inicio de Lilian
El inicio del rango de fechas Lilian es el viernes 15 de octubre de 1582, fecha de adopción del calendario gregoriano. Las fechas lilianas anteriores a esta fecha no están definidas. Por lo tanto:

- El día cero es 00:00:00 14 de octubre de 1582.
- El primer día es 00:00:00 15 de octubre de 1582.

Todas las fechas de entrada válidas deben ser posteriores a 00:00:00 15 de octubre de 1582.

Fecha de finalización de Lilian
La fecha de Lilian final se establece en 31 de diciembre de 9999. Las fechas lililianas posteriores a esta fecha no están definidas porque 9999 es el año de cuatro dígitos más alto posible.

Referencias relacionadas
Apéndice F, “Mensajes de tiempo de ejecución”, en la página 625

Términos y series de caracteres de imagen

Utilice *series de imágenes* (plantillas que indican el formato de los datos de entrada o el formato necesario de los datos de salida) para varios de los servicios invocables de Fecha y hora .

Tabla 49. ***Términos y series de caracteres de imagen***

Términos de imagen	Explicaciones	Valores válidos	Notas
Y	Año de un dígito	0-9	Y válido sólo para salida. YY asume el rango establecido por CEESCEN. YYY/ZYY se utiliza con <JJJJ>, <CCCC>y <CCCCCCCC>.
YY	Año de dos dígitos	00-99	
YYY	Año de tres dígitos	000-999	
ZYY	Año de tres dígitos dentro de la era	1-999	
YYYY	Año de cuatro dígitos	1582-9999	

Tabla 49. ***Términos y series de caracteres de imagen*** *(continuación)*

Términos de imagen	Explicaciones	Valores válidos	Notas
<JJJJ>	Nombre de la era japonesa en caracteres Kanji con codificación hexadecimal UTF-16	Reiwa (NX'E44E8C54')	Afecta al campo YY : si se especifica <JJJJ> , YY significa el año dentro de la era japonesa. Por ejemplo, 1988 equivale a Showa 63.
		Heisei (NX'735E1062')	
		Showa (NX'2D668C54')	
		Taisho (NX'2759636B')	
		Meiji (NX'0E66BB6C')	
MM	Mes de dos dígitos	01-12	Para la salida, cero inicial suprimido. Para entrada, ZM se trata como MM.
ZM	Mes de uno o dos dígitos	1-12	
RRRR	Mes numérico romano	I*bbb*-XII*b* (justificado a la izquierda)	Para la entrada, la serie de origen se convierte en mayúsculas. Para la salida, sólo en mayúsculas. I=Ene, II=Feb, ..., XII=Dic.
RRRZ			
MMM	Mes de tres caracteres, mayúsculas	JAN-DEC	Para la entrada, la serie de origen siempre se convierte en mayúsculas. Para la salida, M genera mayúsculas y m genera minúsculas. La salida se rellena con espacios en blanco (*b*) (a menos que se especifique Z) o se trunca para que coincida con el número de Ms, hasta 20.
Mmm	Mes de tres caracteres, mayúsculas y minúsculas	Ene-Dic	
MMMM...M	3–mes de 20 caracteres, mayúsculas	ENERO*bb*-DECEMBER*b*	
Mmmm...m	3–Mes de 20 caracteres, mayúsculas y minúsculas	Enero*bb*-diciembre*b*	
MMMMMMMMMZ	Blancos de cola suprimidos	ENERO-DICIEMBRE	
Mmmmmmmmmz	Blancos de cola suprimidos	Enero-Diciembre	
DD	Día del mes de dos dígitos	01-31	Para la salida, el cero inicial siempre se suprime. Para entrada, ZD se trata como DD.
ZD	Día del mes de uno o dos dígitos	1-31	
DDD	Día del año (día juliano)	001-366	
HH	Hora de dos dígitos	00-23	Para la salida, cero inicial suprimido. Para entrada, ZH se trata como HH. Si se especifica AP , los valores válidos son 01-12.
ZH	Hora de uno o dos dígitos	0-23	
MI	Minutos	00-59	
SS	Segundo		
9	Décimas de segundo	0-9	Sin redondeo
99	Centésimas de segundo	00-99	
999	Milésimas de segundo	000-999	

Tabla 49. ***Términos y series de caracteres de imagen*** *(continuación)*			
Términos de imagen	**Explicaciones**	**Valores válidos**	**Notas**
AP	Indicador AM/PM	AM o PM	AP afecta al campo HH/ZH . Para la entrada, la serie de origen siempre se convierte en mayúsculas. Para la salida, AP genera mayúsculas y ap genera minúsculas.
ap		am o pm	
A.P.		A.M. o P.M.	
a.p.		A.m. o p.m.	
W	Día de la semana de un carácter	S, M, T, W, T, F, S	Para la entrada, los Ws se ignoran. Para la salida, W genera mayúsculas y w genera minúsculas. Salida rellenada con espacios en blanco (a menos que se especifique Z) o truncada para que coincida con el número de Ws, hasta 20.
WWW	Día de tres caracteres, en mayúsculas	SUN-SAT	
Www	Día de tres caracteres, mayúsculas y minúsculas	Sol-Sáb	
WWW...W	3–Día de 20 caracteres, mayúsculas	DOMINGO*bbb*-SÁBADO*b*	
Www...w	3–Día de 20 caracteres, mayúsculas y minúsculas	Domingo*bbb*-sábado*b*	
WWWWWWWWWWZ	Blancos de cola suprimidos	DOMINGO-SÁBADO	
Wwwwwwwwwz	Blancos de cola suprimidos	Domingo-sábado	
Todos los demás	Delimitadores	X'01 '-X'FF' (X'00 ' está reservado para uso interno por parte de los servicios invocables de Fecha y hora .)	Para la entrada, se trata como delimitadores entre el mes, día, año, hora, minuto, segundo y fracción de segundo. Para la salida, se copia exactamente como está en la serie de destino.
Nota: los caracteres en blanco se indican mediante el símbolo *b*.			

La tabla siguiente define los Eras en japonés utilizados por los servicios de fecha y hora cuando se especifica < JJJJ >.

Tabla 50. ***Errores en japonés***			
Primera fecha de la era japonesa	**Nombre de era**	**Nombre de era en Kanji con codificación hexadecimal UTF-16**	**Valores de año válidos**
1868-09-08	Meiji	NX'0E66BB6C'	01-45
1912-07-30	Taisho	NX'2759636B'	01-15
1926-12-25	Mostrar	NX'2D668C54'	01-64
1989-01-08	Novillas	NX'735E1062'	01-31
2019-05-01	Reiwa	NX'E44E8C54'	01-999 (01 = 2019)

"Ejemplo: series de imágenes de fecha y hora" en la página 545

Ejemplo: series de imágenes de fecha y hora

Estos son ejemplos de series de imágenes reconocidas por los servicios de fecha y hora.

Tabla 51. ***Ejemplos de series de imágenes de fecha y hora***

Series de imágenes	Ejemplos	Comentarios
YYMMDD	880516	
YYYYMMDD	19880516	
YYYY-MM-DD	1988-05-16	1988-5-16 también sería una entrada válida.
<JJJJ> YY.MM.DD	Showa 63,05.16	*Showa* es un nombre de la era japonesa. Showa 63 es igual a 1988.
MMDDYY	050688	El formato de año de un dígito (Y) sólo es válido para la salida.
MM/DD/YY	05/06/88	
ZM/ZD/YY	5/6/88	
MM/DD/YYYY	05/06/1988	
MM/DD/Y	05/06/8	
DD.MM.YY	09.06.88	Z suprime los ceros y blancos.
DD-RRRR-YY	09-VI -88	
DD MMM YY	09 JUN 88	
DD Mmmmmmmmm YY	09 junio 88	
ZD Mmmmmmmmmz YY	9 junio 88	
Mmmmmmmmmz ZD, YYYY	9 de junio de 1988	
ZDMMMMMMMMzYY	9JUNE88	
YY.DDD	88.137	Fecha juliana
YYDDD	88137	
YYYY/DDD	1988/137	
YYMMDDHHMISS	880516204229	Indicación de fecha y hora: válida solo para CEESECS y CEEDATM. Si se utiliza con CEEDATE, las posiciones de tiempo se llenan con ceros. Si se utiliza con los campos CEEDAYS, HH, MI, SSy 999 se ignoran.
YYYYMMDDHHMISS	19880516204229	
YYYY-MM-DD HH:MI:SS.999	1988-05-16 20:42:29.046	
WWW, ZM/ZD/YY HH:MI AP	MON, 5/16/88 08:42 PM	
Wwwwwwwwwz, DD Mmm YYYY, ZH:MI AP	Lunes, 16 de mayo de 1988, 8:42 PM	
Nota: Los caracteres en minúsculas sólo se pueden utilizar para términos de imagen alfabéticos.		

Ventana de siglo

Para procesar años de dos dígitos en el año 2000 y posteriores, los servicios invocables de Fecha y hora utilizan un esquema deslizante en el que se supone que todos los años de dos dígitos se encuentran dentro de un intervalo de 100 años (la *ventana del siglo*) que empieza 80 años antes de la fecha actual del sistema.

En el año 2010, por ejemplo, los 100 años que van de 1930 a 2029 son la ventana de siglo predeterminada para los servicios invocables de Fecha y hora . Así en 2010, los años 30 a 99 son reconocidos como 1930-1999, y los años 00 a 29 son reconocidos como 2000-2029.

Para el año 2080, todos los años de dos dígitos se reconocerán como 20*nn*. En 2081, 00 serán reconocidos como año 2100.

Es posible que algunas aplicaciones necesiten configurar un intervalo de 100 años diferente. Por ejemplo, los bancos a menudo manejan bonos a 30 años, que podrían ser vencidos 01/31/30. El año 30 de dos dígitos sería reconocido como el año 1930 si la ventana del siglo descrita anteriormente estuviera en vigor.

El servicio invocable CEESCEN le permite cambiar la ventana del siglo. Un servicio complementario, CEEQCEN, consulta la ventana del siglo actual.

Puede utilizar CEEQCEN y CEESCEN, por ejemplo, para hacer que una subrutina utilice un intervalo diferente para el proceso de fecha que el utilizado por su rutina padre. Antes de volver, la subrutina debe restablecer el intervalo a su valor anterior.

"Ejemplo: consultar y cambiar la ventana del siglo" en la página 546

Ejemplo: consultar y cambiar la ventana del siglo

El ejemplo siguiente muestra cómo consultar, establecer y restaurar el punto de partida de la ventana del siglo utilizando los servicios CEEQCEN y CEESCEN.

El ejemplo llama a CEEQCEN para obtener un entero (OLDCEN) que indica cuántos años antes comenzó la ventana del siglo actual. A continuación, cambia temporalmente el punto de partida de la ventana del siglo actual a un nuevo valor (TEMPCEN) llamando a CEESCEN con ese valor. Puesto que la ventana de siglo se establece en 30, se supone que los años de dos dígitos que siguen a la llamada CEESCEN se encuentran dentro del intervalo de 100 años que empieza 30 años antes de la fecha actual del sistema.

Por último, después de procesar fechas (no mostradas) utilizando la ventana de siglo temporal, el ejemplo vuelve a llamar a CEESCEN para restablecer el punto de partida de la ventana de siglo a su valor original.

```
 WORKING-STORAGE SECTION.
 77 OLDCEN  PIC S9(9) COMP.
 77 TEMPCEN PIC S9(9) COMP.
 77 QCENFC  PIC X(12).
 . . .
 77 SCENFC1 PIC X(12).
 77 SCENFC2 PIC X(12).
 . . .
 PROCEDURE DIVISION.
     . . .
** Call CEEQCEN to retrieve and save current century window
     CALL "CEEQCEN" USING OLDCEN, QCENFC.
** Call CEESCEN to temporarily change century window to 30
     MOVE 30 TO TEMPCEN.
     CALL "CEESCEN" USING TEMPCEN, SCENFC1.
** Perform date processing with two-digit years
     . . .
** Call CEESCEN again to reset century window
     CALL "CEESCEN" USING OLDCEN, SCENFC2.
     . . .
     GOBACK.
```

Referencias relacionadas
Apéndice C, "Servicios invocables de fecha y hora", en la página 561

Utilización de la sentencia SORT de formato 2 para ordenar una tabla

Se recomienda utilizar la sentencia SORT de formato 2 para ordenar una tabla. Proporciona las ventajas siguientes cuando se compara con la sentencia SORT de formato 1.

Tabla 52. Comparación de sentencias SORT de formato 1 y formato 2

Características	**Sentencias SORT de formato 1**	**Sentencias SORT de formato 2**
Se puede utilizar para ordenar un archivo o una tabla	Sí	No, es sólo para tablas
Requiere DFSORT o un programa de ordenación equivalente	Sí	No
Soportado en CICS	Limitado	Sí
Soportado en UNIX System Services	No	Sí
La tabla se puede ordenar utilizando una sola sentencia SORT , lo que simplifica la codificación	No, requiere las cláusulas SELECT , las entradas SD con descripciones de registro y los procedimientos de entrada y salida	Sí
Las claves para la ordenación se pueden especificar como parte de la definición de tabla, que también se puede utilizar en la sentencia SEARCH ALL	No, las claves deben especificarse en la sentencia SORT . Si la tabla debe buscarse también utilizando SEARCH ALL , las claves también deben especificarse de forma redundante como parte de la definición de tabla.	Sí, y también da soporte a la especificación de claves en la sentencia SORT si es necesario
Puede filtrar o preprocesar elementos de tabla durante el proceso de ordenación	Sí, utilizando procedimientos de entrada y salida	No, todos los elementos de tabla se pasan a SORT tal cual
Utiliza registros especiales que incluyen SORT-CONTROL, SORT-CORE-SIZE, SORT-FILE-SIZE, SORT-MESSAGE, SORT-MODE-SIZEy SORT-RETURN	Sí	No
Se puede ejecutar dentro del rango de un procedimiento de entrada o salida	No	Sí

Nota: No utilice el formato 2 SORT con tablas grandes en un entorno en el que el almacenamiento está restringido, porque el formato 2 SORT utiliza el almacenamiento dinámico para realizar la clasificación.

Referencias relacionadas
Sentencia SORT (*COBOL for Linux en x86 Consulta de lenguaje*)

Apéndice A. Consideraciones sobre el formato de datos de host de IBM Z

La información siguiente de trata sobre las consideraciones, restricciones y limitaciones de que se aplican al uso de la representación interna de datos de host de IBM Z .

Las opciones de compilador CHAR y FLOAT determinan si se utiliza el formato de datos de host IBM Z o el formato de datos nativo (aparte de para elementos COMP-5 o elementos definidos con la frase NATIVE en la cláusula USAGE). (Los términos *formato de datos de host* y *formato de datos nativo* de esta información hacen referencia a la representación interna de los elementos de datos.)

Acceso CICS

No hay soporte de formato de datos de host IBM Z para programas COBOL convertidos por el conversor independiente o integrado CICS y ejecutados en TXSeries o CICS TX.

Servicios invocables de fecha y hora

Puede utilizar los servicios invocables de Fecha y hora con las representaciones internas de formato de datos de host de IBM Z . Todos los parámetros pasados a los servicios invocables deben estar en formato de datos de host IBM Z . No puede mezclar representaciones internas de datos nativas y de host en la misma llamada a un servicio Fecha y hora .

Excepciones de desbordamiento de coma flotante

Debido a las diferencias en los límites de las representaciones de datos de coma flotante en la estación de trabajo de Linux y el host de IBM Z , es posible si FLOAT(BE) está en vigor que se pueda producir una excepción de desbordamiento de coma flotante durante la conversión entre los dos formatos. Por ejemplo, puede recibir el siguiente mensaje en la estación de trabajo cuando ejecute un programa que se ejecute correctamente en el host:

```
IWZ053S An overflow occurred on conversion to floating point
```

Para evitar este problema, debe tener en cuenta los valores máximos de coma flotante soportados en cada plataforma para los tipos de datos respectivos. Los límites se muestran en la tabla siguiente.

Tabla 53. ***Valores máximos de coma flotante***

Tipo de datos	Valor máximo de estación de trabajo	Valor máximo de host de IBM Z
COMP-1	*(2 * *128-2 * * 4) (aprox.*3,4028E+38)	*(16 * *63-16 * *57) (aprox.*7,2370E+75)
COMP-2	*(2 * *1024-2 * *971) (aprox.*1.7977E+308)	*(16 * *63-16 * *49) (aprox.*7,2370E+75)
* Indica que el valor puede ser positivo o negativo.		

Como se muestra anteriormente, el host puede transportar un valor COMP-1 mayor que la estación de trabajo y la estación de trabajo puede transportar un valor COMP-2 mayor que el host.

Db2

Las opciones de compilador de formato de datos de host de IBM Z se pueden utilizar con programas Db2 .

Aplicaciones Distributed Computing Environment

Las opciones de compilador de formato de datos de host de IBM Z no se deben utilizar con programas Distributed Computing Environment .

Datos de archivo

- Los datos EBCDIC y los datos binarios hexadecimales se pueden leer y escribir en cualquier archivo secuencial, relativo o indexado. No se lleva a cabo ninguna conversión automática.
- Si está accediendo a archivos que contienen datos de host, utilice las opciones de compilador `BINARY(BE)`, `COLLSEQ(EBCDIC)`, `CHAR(EBCDIC)`, y `FLOAT(BE)` para procesar datos binarios, datos de caracteres EBCDIC y datos hexadecimales de coma flotante que se adquieren de estos archivos.

SORT

Todos los IBM Z formatos de datos de host excepto DBCS (`USAGE DISPLAY-1`) se pueden utilizar como claves de clasificación.

Conceptos relacionados
"Formatos para datos numéricos" en la página 39

Referencias relacionadas
"opciones de compilador" en la página 265

Apéndice B. Resultados intermedios y precisión aritmética

El compilador maneja las sentencias aritméticas como una sucesión de operaciones realizadas de acuerdo con la prioridad del operador y configura campos intermedios para que contengan los resultados de dichas operaciones. El compilador utiliza algoritmos para determinar el número de enteros y decimales que se deben reservar.

Los resultados intermedios son posibles en los siguientes casos:

- En una sentencia ADD o SUBTRACT que contiene más de un operando inmediatamente después del verbo
- En una sentencia COMPUTE que especifica una serie de operaciones aritméticas o varios campos de resultados
- En una expresión aritmética contenida en una sentencia condicional o en una especificación de modificación de referencia
- En una sentencia ADD, SUBTRACT, MULTIPLYo DIVIDE que utiliza la opción GIVING y varios campos de resultados
- En una sentencia que utiliza una función intrínseca como operando

"Ejemplo: cálculo de resultados intermedios" en la página 553

La precisión de los resultados intermedios depende de si compila utilizando la opción predeterminada ARITH(COMPAT) (denominada *modalidad de compatibilidad*) o utilizando ARITH(EXTEND) (denominada *modalidad ampliada*).

En la modalidad de compatibilidad, la evaluación de las operaciones aritméticas no cambia de la de IBM Conjunto COBOL para Linux:

- Se utiliza un máximo de 30 dígitos para los resultados intermedios de punto fijo.
- Las funciones intrínsecas de coma flotante devuelven resultados de coma flotante de precisión larga (64 bits).
- Las expresiones que contienen operandos de coma flotante, exponentes fraccionales o funciones intrínsecas de coma flotante se evalúan como si todos los operandos que no están en coma flotante se convirtieran a coma flotante de precisión larga y las operaciones de coma flotante se utilizaran para evaluar la expresión.
- Los literales de coma flotante y los elementos de datos de coma flotante externos se convierten en coma flotante de precisión larga para su proceso.

En el modo ampliado, la evaluación de las operaciones aritméticas tiene las características siguientes:

- Se utiliza un máximo de 31 dígitos para los resultados intermedios de punto fijo.
- Las funciones intrínsecas de coma flotante devuelven resultados de coma flotante de precisión ampliada (128 bits) .
- Las expresiones que contienen operandos de coma flotante, exponentes fraccionales o funciones intrínsecas de coma flotante se evalúan como si todos los operandos que no están en coma flotante se convirtieran a coma flotante de precisión ampliada y las operaciones de coma flotante se utilizaran para evaluar la expresión.
- Los literales de coma flotante y los elementos de datos de coma flotante externos se convierten en coma flotante de precisión ampliada para su proceso.

Conceptos relacionados

"Formatos para datos numéricos" en la página 39
"Punto fijo contrastado con aritmética de coma flotante" en la página 54

Referencias relacionadas
“Datos de punto fijo y resultados intermedios” en la página 553
“Datos de coma flotante y resultados intermedios” en la página 558
“Expresiones aritméticas en sentencias no aritméticas” en la página 559
“HARIT” en la página 270

Terminología utilizada para resultados intermedios

Para comprender esta información sobre los resultados intermedios, debe comprender la terminología siguiente.

i
Número de posiciones enteras transportadas para un resultado intermedio. (Si utiliza la frase `ROUNDED` , es posible que se lleve un número entero más para la precisión si es necesario.)

d
Número de posiciones decimales transportadas para un resultado intermedio. (Si utiliza la frase `ROUNDED` , es posible que se lleve una posición decimal más para la precisión si es necesario.)

máx. dd
En una sentencia determinada, el mayor de los elementos siguientes:

- El número de posiciones decimales necesarias para el campo o campos de resultado final
- El número máximo de posiciones decimales definidas para cualquier operando, excepto divisores o exponentes
- El *outer-dmax* para cualquier operando de función

máx-dmáx-interno
En referencia a una función, el mayor de los elementos siguientes:

- El número de posiciones decimales definidas para cualquiera de sus argumentos elementales
- El *dmax* para cualquiera de sus argumentos de expresión aritmética
- El *outer-dmax* para cualquiera de sus funciones incorporadas

máx-dexterno
Número de posiciones decimales que un resultado de función contribuye a operaciones fuera de su propia evaluación (por ejemplo, si la función es un operando en una expresión aritmética o un argumento en otra función).

op1
El primer operando de una sentencia aritmética generada (en división, el divisor).

op2
El segundo operando de una sentencia aritmética generada (en división, el dividendo).

i1 , i2
El número de lugares enteros en *op1* y *op2*, respectivamente.

d1 , d2
El número de posiciones decimales en *op1* y *op2*, respectivamente.

ir
El resultado intermedio cuando se realiza una operación o sentencia aritmética generada. (Los resultados intermedios se generan en registros o en ubicaciones de almacenamiento.)

ir1 , ir2
Resultados intermedios sucesivos. (Los resultados intermedios sucesivos pueden tener la misma ubicación de almacenamiento.)

Referencias relacionadas
Frase REDONDEADA (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: cálculo de resultados intermedios

El ejemplo siguiente muestra cómo el compilador realiza una sentencia aritmética como una sucesión de operaciones, almacenando los resultados intermedios según sea necesario.

```
COMPUTE Y = A + B * C - D / E + F ** G
```

El resultado se calcula en el orden siguiente:

1. Exponencie F G generando *ir1*.
2. Multiplique B por C produciendo *ir2*.
3. Divida E en D produciendo *ir3*.
4. Añada A a `ir2` generando *ir4*.
5. Reste *ir3* de *ir4* generando *ir5*.
6. Añada *ir5* a *ir1* produciendo Y.

Tareas relacionadas
“Utilización de expresiones aritméticas” en la página 50

Referencias relacionadas
“Terminología utilizada para resultados intermedios” en la página 552

Datos de punto fijo y resultados intermedios

El compilador determina el número de posiciones enteras y decimales en un resultado intermedio.

Suma, resta, multiplicación y división

La tabla siguiente muestra la precisión teóricamente posible como resultado de la suma, la resta, la multiplicación o la división.

Operación	Posiciones enteras	Posiciones decimales
+ o-	(*i1* o *i2*) + 1, el que sea mayor	*d1* o *d2*, el que sea mayor
*	*i1* + *i2*	*d1* + *d2*
/	*i2* + *d1*	(*d2* - *d1*) o *dmax*, el que sea mayor

Debe definir los operandos de cualquier sentencia aritmética con suficientes posiciones decimales para obtener la precisión que desea en el resultado final.

La tabla siguiente muestra el número de lugares que el compilador lleva para los resultados intermedios de punto fijo de las operaciones aritméticas que implican suma, resta, multiplicación o división en *modalidad de compatibilidad* (es decir, cuando la opción de compilador predeterminada `ARITH(COMPAT)` está en vigor):

Valor de *i* + *d*	Valor de *d*	Valor de *i* + *dmax*	Número de plazas transportadas para *ir*
< 30 o = 30	Cualquier valor	Cualquier valor	*i* entero y *d* posiciones decimales
>30	<*dmax* o =*dmax*	Cualquier valor	30-*d* entero y *d* posiciones decimales
	>*máx. dd*	< 30 o = 30	*i* entero y 30-*i* posiciones decimales
		>30	30-*dmax* entero y *dmax* posiciones decimales

La tabla siguiente muestra el número de lugares que el compilador lleva para los resultados intermedios de punto fijo de las operaciones aritméticas que implican suma, resta, multiplicación o división en *modalidad ampliada* (es decir, cuando la opción de compilador `ARITH(EXTEND)` está en vigor):

Valor de *i* + *d*	**Valor de *d***	**Valor de *i* + *dmax***	**Número de plazas transportadas para *ir***
< 31 o = 31	Cualquier valor	Cualquier valor	*i* entero y *d* posiciones decimales
>31	<*dmax* o =*dmax*	Cualquier valor	31-*d* entero y *d* posiciones decimales
	>*máx. dd*	< 31 o = 31	*i* entero y 31-*i* posiciones decimales
		>31	31-*dmax* entero y *dmax* posiciones decimales

Exponenciación

La exponenciación se representa mediante la expresión *op1* ** *op2*. Basándose en las características de *op2*, el compilador maneja la exponenciación de números de punto fijo de una de estas tres maneras:

- Cuando *op2* se expresa con decimales, se utilizan instrucciones de coma flotante.
- Cuando *op2* es un literal o constante integral, el valor *d* se calcula como

```
d = d1 * |op2|
```

y el valor *i* se calcula basándose en las características de *op1*:

– Cuando *op1* es un nombre de datos o una variable,

```
i = i1 * |op2|
```

– Cuando *op1* es un literal o constante, *i* se establece igual al número de enteros en el valor de *op1* ** |*op2* |.

En modalidad de compatibilidad (compilación utilizando `ARITH(COMPAT)`), el compilador que ha calculado *i* y *d* realiza la acción indicada en la tabla siguiente para manejar los resultados intermedios *ir* de la exponenciación.

Valor de *i* + *d*	**Otras condiciones**	**Acción emprendida**
<30	any	*i* entero y *d* posiciones decimales se transportan para *ir*.
=30	*op1* tiene un número impar de dígitos.	*i* entero y *d* posiciones decimales se transportan para *ir*.
	op1 tiene un número par de dígitos.	La misma acción que cuando *op2* es un nombre de datos integral o una variable (que se muestra a continuación). Excepción: para un entero de 30 dígitos elevado a la potencia del literal 1, *i* entero y *d* posiciones decimales se transportan para *ir*.
>30	any	La misma acción que cuando *op2* es un nombre de datos integral o una variable (se muestra a continuación)

En modalidad ampliada (compilación utilizando `ARITH(EXTEND)`), el compilador que ha calculado *i* y *d* realiza la acción indicada en la tabla siguiente para manejar los resultados intermedios *ir* de la exponenciación.

Valor de *i* + *d*	Otras condiciones	Acción emprendida
<31	any	*i* entero y *d* posiciones decimales se transportan para *ir*.
= 31 o > 31	any	La misma acción que cuando *op2* es un nombre de datos integral o una variable (que se muestra a continuación). Excepción: para un entero de 31 dígitos elevado a la potencia del literal 1, se transportan *i* enteros y *d* posiciones decimales para *ir*.

Si *op2* es negativo, el valor de 1 se divide por el resultado generado por el cálculo preliminar. Los valores de *i* y *d* que se utilizan se calculan siguiendo las reglas de división para los datos de punto fijo que ya se muestran anteriormente.

- Cuando *op2* es un nombre de datos o variable integral, Se utilizan *dmax* decimales y 30-*dmax* (modalidad de compatibilidad) o 31-*dmax* (modalidad ampliada) enteros. *op1* se multiplica por sí mismo (|*op2*|-1) veces para un valor distinto de cero *op2*.

  Si *op2* es igual a 0, el resultado es 1. Se aplican las condiciones Division-by-0 y exponenciación SIZE ERROR .

Los exponentes de punto fijo con más de nueve dígitos significativos siempre se truncan a nueve dígitos. Si el exponente es un literal o constante, se emite un mensaje de diagnóstico de compilador de nivel E; de lo contrario, se emite un mensaje informativo en tiempo de ejecución.

"Ejemplo: exponenciación en aritmética de punto fijo" en la página 555

Referencias relacionadas
"Terminología utilizada para resultados intermedios" en la página 552
"Resultados intermedios truncados" en la página 556
"Datos binarios y resultados intermedios" en la página 556
"Datos de coma flotante y resultados intermedios" en la página 558
"Funciones intrínsecas evaluadas en aritmética de punto fijo" en la página 556
"HARIT" en la página 270
Frases de ERROR TAMAÑO (*COBOL for Linux en x86 Consulta de lenguaje*)

Ejemplo: exponenciación en aritmética de punto fijo

El ejemplo siguiente muestra cómo el compilador realiza una exponenciación a una potencia entera distinta de cero como una sucesión de multiplicaciones, almacenando los resultados intermedios según sea necesario.

```
COMPUTE Y = A ** B
```

Si B es igual a 4, el resultado se calcula como se muestra a continuación. Los valores de *i* y *d* que se utilizan se calculan de acuerdo con las reglas de multiplicación para datos de punto fijo y resultados intermedios (a los que se hace referencia a continuación).

1. Multiplique A por A produciendo *ir1*.
2. Multiplique *ir1* por A generando *ir2*.
3. Multiplique *ir2* por A produciendo *ir3*.
4. Mueva *ir3* a *ir4*.

   ir4 tiene *dmax* posiciones decimales. Puesto que B es positivo, *ir4* se mueve a Y. Sin embargo, si B fuera igual a -4, se realizaría un quinto paso adicional:
5. Divida *ir4* en 1, lo que genera *ir5*.

ir5 tiene *dmax* posiciones decimales y, a continuación, se movería a Y.

Referencias relacionadas
“Terminología utilizada para resultados intermedios” en la página 552
“Datos de punto fijo y resultados intermedios” en la página 553

Resultados intermedios truncados

Siempre que el número de dígitos en un resultado intermedio excede 30 en modalidad de compatibilidad o 31 en modalidad ampliada, el compilador se trunca a 30 (modalidad de compatibilidad) o 31 (modalidad ampliada) dígitos y emite un aviso. Si se produce un truncamiento en tiempo de ejecución, se emite un mensaje y el programa continúa ejecutándose.

Si desea evitar el truncamiento de resultados intermedios que pueden producirse en cálculos de punto fijo, utilice en su lugar operandos de coma flotante (`COMP-1` o `COMP-2`).

Conceptos relacionados
“Formatos para datos numéricos” en la página 39

Referencias relacionadas
“Datos de punto fijo y resultados intermedios” en la página 553
“HARIT” en la página 270

Datos binarios y resultados intermedios

Si una operación que implica operandos binarios requiere resultados intermedios de más de 18 dígitos, el compilador convierte los operandos en decimales internos antes de realizar la operación. Si el campo de resultado es binario, el compilador convierte el resultado de decimal interno a binario.

Los operandos binarios son más eficaces cuando los resultados intermedios no superarán los nueve dígitos.

Referencias relacionadas
“Datos de punto fijo y resultados intermedios” en la página 553
“HARIT” en la página 270

Funciones intrínsecas evaluadas en aritmética de punto fijo

El compilador determina los valores *inner-dmax* y *outer-dmax* para una función intrínseca a partir de las características de la función.

Funciones enteras

Las funciones intrínsecas de entero devuelven un entero; por lo tanto, su *outer-dmax* es siempre cero. Para aquellas funciones de enteros cuyos argumentos deben ser enteros, *inner-dmax* es, por lo tanto, siempre cero.

La tabla siguiente resume el *inner-dmax* y la precisión del resultado de la función.

Función	*Interno-dmax*	Precisión de dígito del resultado de la función
`FECHA-DE-ENTERO`	0	8
`DATE-TO-AAAAMMDD`	0	8
`DAY-OF-INTEGER`	0	7
`DAY-TO-AAAADDD`	0	7
`FACTORIAL`	0	30 en modalidad de compatibilidad, 31 en modalidad ampliada
`ENTERO-DE-FECHA`	0	7

Función	*Interno-dmax*	Precisión de dígito del resultado de la función
INTEGER-OF-DAY	0	7
LENGTH	n/d	9
MOD	0	min(*i1 i2*)
SEÑOR	n/d	3
SEÑOR-MAX		9
SEÑOR-MIN		9
YEAR-TO-YYYY	0	4
Entero		Para un argumento de punto fijo: un dígito más que en el argumento. Para un argumento de coma flotante: 30 en modalidad de compatibilidad, 31 en modalidad ampliada.
INTEGER-PARTE		Para un argumento de punto fijo: el mismo número de dígitos que en el argumento. Para un argumento de coma flotante: 30 en modalidad de compatibilidad, 31 en modalidad ampliada.

Funciones mixtas

Una función intrínseca *mixta* es una función cuyo tipo de resultado depende del tipo de sus argumentos. Una función mixta es de punto fijo si todos sus argumentos son numéricos y ninguno de sus argumentos es de coma flotante. (Si cualquier argumento de una función mixta es de coma flotante, la función se evalúa con instrucciones de coma flotante y devuelve un resultado de coma flotante.) Cuando una función mixta se evalúa con aritmética de punto fijo, el resultado es entero si todos los argumentos son enteros; de lo contrario, el resultado es punto fijo.

Para las funciones mixtas MAX, MIN, RANGE, REMy SUM, *outer-dmax* siempre es igual a *inner-dmax* (y ambos son por lo tanto cero si todos los argumentos son enteros). Para determinar la precisión del resultado devuelto para estas funciones, aplique las reglas para la aritmética de punto fijo y los resultados intermedios (como se indica a continuación) a cada paso del algoritmo.

MAX

1. Asigne el primer argumento al resultado de la función.
2. Para cada argumento restante, realice los pasos siguientes:
   a. Compare el valor algebraico del resultado de la función con el argumento.
   b. Asigne el mayor de los dos al resultado de la función.

MIN

1. Asigne el primer argumento al resultado de la función.
2. Para cada argumento restante, realice los pasos siguientes:
   a. Compare el valor algebraico del resultado de la función con el argumento.
   b. Asigne el menor de los dos al resultado de la función.

RANGE

1. Utilice los pasos para MAX para seleccionar el argumento máximo.
2. Utilice los pasos para MIN para seleccionar el argumento mínimo.
3. Reste el argumento mínimo del máximo.
4. Asigne la diferencia al resultado de la función.

REM

1. Divida el argumento uno por el argumento dos.

2. Elimine todos los dígitos no enteros del resultado del paso 1.
3. Multiplique el resultado del paso 2 por el argumento dos.
4. Reste el resultado del paso 3 del argumento uno.
5. Asigne la diferencia al resultado de la función.

SUM

1. Asigne el valor 0 al resultado de la función.
2. Para cada argumento, realice los pasos siguientes:
   a. Añada el argumento al resultado de la función.
   b. Asigne la suma al resultado de la función.

Referencias relacionadas
"Terminología utilizada para resultados intermedios" en la página 552
"Datos de punto fijo y resultados intermedios" en la página 553
"Datos de coma flotante y resultados intermedios" en la página 558
"HARIT" en la página 270

Datos de coma flotante y resultados intermedios

Si alguna operación en una expresión aritmética se calcula en aritmética de coma flotante, toda la expresión se calcula como si todos los operandos se convirtieran en coma flotante y las operaciones se realizaran utilizando instrucciones de coma flotante.

Las instrucciones de coma flotante se utilizan para calcular una expresión aritmética si se cumple alguna de las condiciones siguientes de la expresión:

- Un receptor u operando es `COMP-1`, `COMP-2`, coma flotante externa o un literal de coma flotante.
- Un exponente contiene posiciones decimales.
- Un exponente es una expresión que contiene un operador de división o exponenciación y *dmax* es mayor que cero.
- Una función intrínseca es una función de coma flotante.

En modalidad de compatibilidad, si una expresión se calcula en aritmética de coma flotante, la precisión utilizada para evaluar las operaciones aritméticas se determina de la forma siguiente:

- Se utiliza una precisión única si todos los receptores y operandos son elementos de datos `COMP-1` y la expresión no contiene operaciones de multiplicación o exponenciación.
- En todos los demás casos, se utiliza una precisión larga.

Siempre que se utiliza la coma flotante de precisión larga para una operación en una expresión aritmética, todas las operaciones de la expresión se calculan como si se utilizaran instrucciones de coma flotante larga.

En la modalidad ampliada, si una expresión se calcula en aritmética de coma flotante, la precisión utilizada para evaluar las operaciones aritméticas se determina de la forma siguiente:

- Se utiliza una precisión única si todos los receptores y operandos son elementos de datos `COMP-1` y la expresión no contiene operaciones de multiplicación o exponenciación.
- La precisión larga se utiliza si todos los receptores y operandos son elementos de datos `COMP-1` o `COMP-2` , al menos un receptor u operando es un elemento de datos `COMP-2` y la expresión no contiene operaciones de multiplicación o exponenciación.
- En todos los demás casos, se utiliza una precisión ampliada.

Siempre que se utiliza la coma flotante de precisión ampliada para una operación en una expresión aritmética, todas las operaciones de la expresión se calculan como si se utilizaran instrucciones de coma flotante de precisión ampliada.

Alerta: Si una operación de coma flotante tiene un campo de resultado intermedio en el que se produce un desbordamiento de exponente, el trabajo termina de forma anómala.

Exponencias evaluadas en aritmética de coma flotante

En la modalidad de compatibilidad, las exponencias de coma flotante siempre se evalúan utilizando la aritmética de coma flotante larga. En la modalidad ampliada, las exponencias de coma flotante siempre se evalúan utilizando la aritmética de coma flotante de precisión ampliada.

El valor de un número negativo elevado a una potencia fraccional no está definido en COBOL. Por ejemplo, (-2) ** 3 es igual a -8, pero (-2) * * (3.000001) no está definido. Cuando una exponenciación se evalúa en coma flotante y existe la posibilidad de que el resultado no esté definido, el exponente se evalúa en tiempo de ejecución para determinar si tiene un valor integral. Si no es así, se emite un mensaje de diagnóstico.

Funciones intrínsecas evaluadas en aritmética de coma flotante

En modalidad de compatibilidad, las funciones intrínsecas de coma flotante siempre devuelven un valor de coma flotante largo (64 bits). En la modalidad ampliada, las funciones intrínsecas de coma flotante siempre devuelven un valor de coma flotante de precisión ampliada (128 bits) .

Las funciones mixtas que tienen al menos un argumento de coma flotante se evalúan utilizando la aritmética de coma flotante.

Referencias relacionadas
“Terminología utilizada para resultados intermedios” en la página 552
“HARIT” en la página 270

Expresiones aritméticas en sentencias no aritméticas

Las expresiones aritméticas pueden aparecer en contextos distintos de las sentencias aritméticas. Por ejemplo, puede utilizar una expresión aritmética con la sentencia IF o EVALUATE .

En tales sentencias, se aplican las reglas para los resultados intermedios con datos de punto fijo y para los resultados intermedios con datos de coma flotante, con los cambios siguientes:

- Las sentencias IF abreviadas se manejan como si las sentencias no se abreviaran.
- En una condición de relación explícita donde al menos una de las comparaciones es una expresión aritmética, *dmax* es el número máximo de posiciones decimales para cualquier operando de cualquiera de las comparaciones, excluyendo los divisores y exponentes. Las reglas para la aritmética de coma flotante se aplican si se cumple alguna de las condiciones siguientes:
  - Cualquier operando en comparand es COMP-1, COMP-2, coma flotante externa o un literal de coma flotante.
  - Un exponente contiene posiciones decimales.
  - Un exponente es una expresión que contiene un operador de división o exponenciación y *dmax* es mayor que cero.

  Por ejemplo:

  ```
  IF operand-1 = expression-1 THEN . . .
  ```

  Si *operand-1* es un nombre de datos definido como COMP-2, las reglas para la aritmética de coma flotante se aplican a *expression-1* incluso si contiene sólo operandos de punto fijo, porque se está comparando con un operando de coma flotante.
- Cuando la comparación entre una expresión aritmética y otro elemento de datos o expresión aritmética no utiliza un operador relacional (es decir, no hay ninguna condición de relación explícita), la expresión aritmética se evalúa sin tener en cuenta los atributos de su comparación. Por ejemplo:

```
EVALUATE expression-1
  WHEN expression-2 THRU expression-3
  WHEN expression-4
  . . .
END-EVALUATE
```

En la sentencia anterior, cada expresión aritmética se evalúa en aritmética de punto fijo o de coma flotante basándose en sus propias características.

Conceptos relacionados
“Punto fijo contrastado con aritmética de coma flotante” en la página 54

Referencias relacionadas
“Terminología utilizada para resultados intermedios” en la página 552
“Datos de punto fijo y resultados intermedios” en la página 553
“Datos de coma flotante y resultados intermedios” en la página 558
sentencia IF (*COBOL for Linux en x86 Consulta de lenguaje*)
sentencia EVALUATE (*COBOL for Linux en x86 Consulta de lenguaje*)
Expresiones condicionales (*COBOL for Linux en x86 Consulta de lenguaje*)

Apéndice C. Servicios invocables de fecha y hora

Utilizando los servicios invocables de Fecha y hora , puede obtener la fecha y hora local actual en varios formatos y puede convertir fechas y horas.

Los servicios invocables de Fecha y hora disponibles se muestran a continuación. Dos de los servicios, CEEQCEN y CEESCEN, proporcionan una forma predecible de manejar años de dos dígitos, como 91 para 1991 o 10 para 2010.

Tabla 54. ***Servicios invocables de fecha y hora***

Servicio invocable	**Descripción**
“CEECBLDY: convertir fecha a formato entero COBOL” en la página 563	Convierte el valor de fecha de tipo carácter al formato de fecha de entero COBOL. El primer día es el 01 de enero de 1601, y el valor se incrementa en uno para cada día posterior.
“CEEDATE: convertir fecha liliana a formato de caracteres” en la página 566	Convierte las fechas en el formato Lilian de nuevo a valores de caracteres.
“CEEDATM: convertir segundos en indicación de fecha y hora de tipo carácter” en la página 570	Convierte el número de segundos en una indicación de fecha y hora de carácter.
“CEEDAYS: convertir fecha a formato Lilian” en la página 573	Convierte los valores de fecha de caracteres al formato Lilian. El primer día es el 15 de octubre de 1582 y el valor se incrementa en uno para cada día posterior.
“CEEDYWK: calcular día de la semana a partir de la fecha lililiana” en la página 576	Proporciona el cálculo del día de la semana.
“CEEGMT: obtener hora media de Greenwich actual” en la página 578	Obtiene la hora media de Greenwich actual (fecha y hora).
“CEEGMTO: obtener desplazamiento de la hora media de Greenwich a la hora local” en la página 580	Obtiene la diferencia entre la hora media de Greenwich y la hora local.
“CEEISEC: convertir enteros en segundos” en la página 582	Convierte el año binario, mes, día, hora, segundo y milisegundo en un número que representa el número de segundos desde 00:00:00 15 de octubre de 1582.
“CEELOCT: obtener fecha u hora local actual” en la página 584	Obtiene la fecha y hora actuales.
“CEEQCEN: consultar la ventana del siglo” en la página 586	Consulta la ventana del siglo de servicios invocables.
“CEESCEN: establecer la ventana del siglo” en la página 587	Establece la ventana de siglo de servicios invocables.
“CEESECI: convertir segundos en enteros” en la página 589	Convierte un número que representa el número de segundos desde 00:00:00 15 de octubre de 1582 en siete enteros binarios separados que representan año, mes, día, hora, minuto, segundo y milisegundo.
“CEESECS: convertir indicación de fecha y hora en segundos” en la página 591	Convierte las indicaciones de fecha y hora de caracteres (una fecha y hora) en el número de segundos desde 00:00:00 15 de octubre de 1582.

*Tabla 54. **Servicios invocables de fecha y hora** (continuación)*	
Servicio invocable	**Descripción**
"CEEUTC: obtener hora universal coordinada" en la página 595	Igual que CEEGMT.
"IGZEDT4: obtener fecha actual" en la página 595	Devuelve la fecha actual con un año de cuatro dígitos con el formato AAMMDD.

Todos estos servicios invocables de Fecha y hora permiten la compatibilidad del código fuente con Enterprise COBOL for z/OS. Sin embargo, existen diferencias significativas en la forma en que se manejan las condiciones.

Los servicios invocables de Fecha y hora son adicionales a las funciones intrínsecas de fecha/hora que se muestran a continuación.

*Tabla 55. **Funciones intrínsecas de fecha y hora***	
Función intrínseca	**Descripción**
CURRENT-DATE	Fecha y hora actuales y diferencia con respecto a la hora media de Greenwich
DATE-OF-INTEGER[1]	Fecha estándar equivalente (AAAAMMDD) de fecha entera
DATE-TO-YYYYMMDD[1]	Fecha estándar equivalente (AAAAMMDD) de fecha entera con un año con ventana, según el intervalo de 100 años especificado
DATEVAL[1]	Campo de fecha equivalente a entero o fecha alfanumérica
DAY-OF-INTEGER[1]	Fecha juliana equivalente (AAAADDD) de fecha entera
DAY-TO-YYYYDDD[1]	Fecha juliana equivalente (AAAAMMDD) de fecha entera con un año con ventana, según el intervalo de 100 años especificado
INTEGER-OF-DATE	Fecha entera equivalente a la fecha estándar (AAAAMMDD)
INTEGER-OF-DAY	Fecha entera equivalente a la fecha juliana (AAAADDD)
UNDATE[1]	No fecha equivalente de entero o campo de fecha alfanumérico
YEAR-TO-YYYY[1]	Equivalente de año ampliado (AAAA) del año con ventana, según el intervalo de 100 años especificado
YEARWINDOW[1]	Año de inicio de la ventana de siglo especificada por la opción de compilador YEARWINDOW
1. El comportamiento depende del valor de la opción de compilador DATEPROC .	

"Ejemplo: formato de fechas para salida" en la página 540

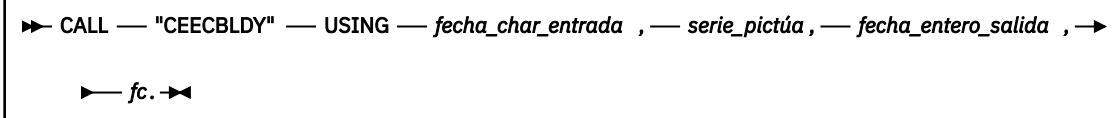
Referencias relacionadas
"Señal de comentarios" en la página 541
sentencia CALL (*COBOL for Linux en x86 Consulta de lenguaje*)
Definiciones de función (*COBOL for Linux en x86 Consulta de lenguaje*)

CEECBLDY: convertir fecha a formato entero COBOL

CEECBLDY convierte una serie que representa una fecha en el número de días desde el 31 de diciembre de 1600. Utilice CEECBLDY para acceder a la ventana de siglo de los servicios invocables de Fecha y hora y para realizar cálculos de fecha con funciones intrínsecas de COBOL.

Este servicio es similar a CEEDAYS excepto que proporciona una serie en formato de entero COBOL, que es compatible con las funciones intrínsecas de COBOL.

Sintaxis de CALL CEECBLDY



***fecha_char_entrada* (entrada)**

Una serie de caracteres con prefijo de longitud de media palabra que representa una indicación de fecha y hora en un formato que se ajusta al especificado por *picture_string*.

La serie de caracteres debe contener entre 5 y 255 caracteres, ambos inclusive. *fecha_char_entrada* puede contener espacios en blanco iniciales o finales. El análisis de una fecha empieza con el primer carácter no en blanco (a menos que la propia serie de imagen contenga espacios en blanco iniciales, en cuyo caso CEECBLDY omite exactamente esa cantidad de posiciones antes de que comience el análisis).

Después de analizar una fecha válida, tal como determina el formato de la fecha especificada en *picture_string*, CEECBLDY ignora todos los caracteres restantes. Las fechas válidas están comprendidas entre el 01 de enero de 1601 y el 31 de diciembre de 9999.

***picture_string* (entrada)**

Una serie de caracteres con prefijo de longitud de media palabra que indica el formato de la fecha especificada en *fecha_char_entrada*.

Cada carácter de *picture_string* corresponde a un carácter de *input_char_date*. Por ejemplo, si especifica MMDDYY como *picture_string*, CEECBLDY lee *input_char_date* de 060288 como 02 de junio de 1988.

Si los delimitadores como la barra inclinada (/) aparecen en la serie de imagen, puede omitir los ceros iniciales. Por ejemplo, las llamadas siguientes a CEECBLDY asignan cada una el mismo valor, 141502 (02 de junio de 1988), a COBINTDTE:

```
MOVE '6/2/88' TO DATEVAL-STRING.
MOVE 6 TO DATEVAL-LENGTH.
MOVE 'MM/DD/YY' TO PICSTR-STRING.
MOVE 8 TO PICSTR-LENGTH.
CALL CEECBLDY USING DATEVAL, PICSTR, COBINTDTE, FC.
```

```
MOVE '06/02/88' TO DATEVAL-STRING.
MOVE 8 TO DATEVAL-LENGTH.
MOVE 'MM/DD/YY' TO PICSTR-STRING.
MOVE 8 TO PICSTR-LENGTH.
CALL CEECBLDY USING DATEVAL, PICSTR, COBINTDTE, FC.
```

```
MOVE '060288' TO DATEVAL-STRING.
MOVE 6 TO DATEVAL-LENGTH.
MOVE 'MMDDYY' TO PICSTR-STRING.
MOVE 6 TO PICSTR-LENGTH.
CALL CEECBLDY USING DATEVAL, PICSTR, COBINTDTE, FC.
```

```
MOVE '88154' TO DATEVAL-STRING.
MOVE 5 TO DATEVAL-LENGTH.
MOVE 'YYDDD' TO PICSTR-STRING.
MOVE 5 TO PICSTR-LENGTH.
CALL CEECBLDY USING DATEVAL, PICSTR, COBINTDTE, FC.
```

Siempre que se incluyan caracteres como, por ejemplo, dos puntos o barras inclinadas en la *picture_string* (como `HH:MI:SS YY/MM/DD`), se contabilizarán como marcadores de posición, pero por lo demás se ignorarán.

Si *picture_string* incluye un símbolo de era japonesa <JJJJ>, la posición YY en *input_char_date* se sustituye por el número de año dentro de la era japonesa. Por ejemplo, el año 1988 equivale al año japonés 63 en la era Showa.

***output_Integer_date* (salida)**

Un entero binario de 32 bits que representa la fecha entera COBOL, el número de días desde el 31 de diciembre de 1600. Por ejemplo, el 16 de mayo de 1988 es el día 141485.

Si *input_char_date* no contiene una fecha válida, *output_Integer_date* se establece en 0 y CEECBLDY termina con un código de retorno simbólico non-CEE000 .

Los cálculos de fecha se realizan fácilmente en *output_Integer_date*, porque *output_Integer_date* es un entero. Las anomalías de año bisiesto y de fin de año no afectan a los cálculos.

***fc* (salida)**

Un código de retroalimentación de 12 bytes (opcional) que indica el resultado de este servicio.

Tabla 56. ***Condiciones simbólicas de CEECBLDY***

Código de comentarios simbólicos	**Gravedad**	**Número de mensaje**	**texto de mensaje**
CEE000	0	--	El servicio se ha completado correctamente.
CEE2EB	3	2507	No se pasaron datos suficientes a CEEDAYS o CEESECS. El valor Lilian no se ha calculado.
CEE2EC	3	2508	El valor de fecha pasado a CEEDAYS o CEESECS no era válido.
CEE2ED	3	2509	La era pasada a CEEDAYS o CEESECS no fue reconocida.
CEE2EH	3	2513	La fecha de entrada pasada en una llamada CEEISEC, CEEDAYS o CEESECS no estaba dentro del rango soportado.
CEE2EL	3	2517	No se ha reconocido el valor de mes en una llamada CEEISEC.
CEE2EM	3	2518	Se ha especificado una serie de imagen no válida en una llamada a un servicio de fecha/hora.
CEE2EO	3	2520	CEEDAYS ha detectado datos no numéricos en un campo numérico, o la serie de fecha no coincidía con la serie de imagen.
CEE2EP	3	2521	El valor de año dentro de era < JJJJ>, < CCCC> o < CCCCCCCCCC> pasado a CEEDAYS o CEESECS era cero.

Notas de uso

- Llame a CEECBLDY sólo desde programas COBOL que utilicen el valor devuelto como entrada para las funciones intrínsecas de COBOL. A diferencia de CEEDAYS, no hay ninguna función inversa de CEECBLDY, porque sólo es para los usuarios de COBOL que desean utilizar el servicio de ventana del

siglo Fecha y hora junto con las funciones intrínsecas de COBOL para los cálculos de fecha. El inverso de CEECBLDY lo proporcionan las funciones intrínsecas DATE-OF-INTEGER y DAY-OF-INTEGER .

- Para realizar cálculos en fechas anteriores al 1 de enero de 1601, añada 4000 al año en cada fecha, convierta las fechas al formato de entero COBOL y, a continuación, realice el cálculo. Si el resultado del cálculo es una fecha, en contraposición a un número de días, convierta el resultado en una serie de fecha y reste 4000 del año.
- De forma predeterminada, los años de dos dígitos se encuentran dentro del rango de 100 años que empieza 80 años antes de la fecha del sistema. Así en 2010, todos los años de dos dígitos representan fechas entre 1930 y 2029, inclusive. Puede cambiar este rango predeterminado utilizando el servicio invocable CEESCEN.

ejemplo

```
************************************************
**                                            **
** Function: Invoke CEECBLDY callable service  **
** to convert date to COBOL integer format.    **
** This service is used when using the         **
** Century Window feature of the Fecha y hora **
** callable services mixed with COBOL          **
** intrinsic functions.                        **
**                                            **
************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. CBLDY.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  CHRDATE.
     02  Vstring-length      PIC S9(4) BINARY.
     02  Vstring-text.
         03  Vstring-char    PIC X
                     OCCURS 0 TO 256 TIMES
                     DEPENDING ON Vstring-length
                         of CHRDATE.
 01  PICSTR.
     02  Vstring-length      PIC S9(4) BINARY.
     02  Vstring-text.
         03  Vstring-char    PIC X
                     OCCURS 0 TO 256 TIMES
                     DEPENDING ON Vstring-length
                         of  PICSTR.
 01  INTEGER                 PIC S9(9) BINARY.
 01  NEWDATE                 PIC 9(8).
 01  FC.
     02  Condition-Token-Value.
     COPY  CEEIGZCT.
         03  Case-1-Condition-ID.
             04  Severity    PIC S9(4) COMP.
             04  Msg-No      PIC S9(4) COMP.
         03  Case-2-Condition-ID
                 REDEFINES Case-1-Condition-ID.
             04  Class-Code  PIC S9(4) COMP.
             04  Cause-Code  PIC S9(4) COMP.
         03  Case-Sev-Ctl    PIC X.
         03  Facility-ID     PIC XXX.
     02  I-S-Info            PIC S9(9) COMP.
*
 PROCEDURE DIVISION.
 PARA-CBLDAYS.
************************************************
** Specify input date and length              **
************************************************
     MOVE 25 TO Vstring-length of CHRDATE.
     MOVE '1 January 00'
         to Vstring-text of CHRDATE.
************************************************
** Specify a picture string that describes     **
** input date, and set the string's length.    **
************************************************
     MOVE 23 TO Vstring-length of PICSTR.
     MOVE 'ZD Mmmmmmmmmmmmmz YY'
               TO Vstring-text of PICSTR.
************************************************
** Call CEECBLDY to convert input date to a    **
```

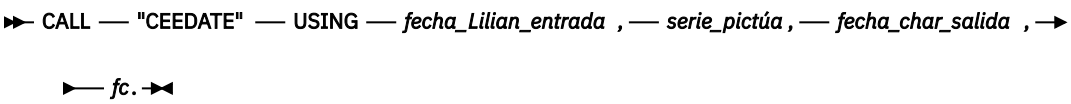
```
** COBOL integer date                          **
************************************************
     CALL 'CEECBLDY' USING CHRDATE, PICSTR,
                           INTEGER, FC.
************************************************
** If CEECBLDY runs successfully, then compute **
**     the date of the 90th day after the      **
**     input date using Intrinsic Functions    **
************************************************
     IF CEE000 of FC  THEN
         COMPUTE INTEGER = INTEGER + 90
         COMPUTE NEWDATE = FUNCTION
             DATE-OF-INTEGER (INTEGER)
         DISPLAY NEWDATE
             ' is Lilian day: ' INTEGER
     ELSE
         DISPLAY 'CEEBLDY failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.
*
     GOBACK.
```

Referencias relacionadas
"Términos y series de caracteres de imagen" en la página 542

CEEDATE: convertir fecha liliana a formato de caracteres

CEEDATE convierte un número que representa una fecha Lilian a una fecha escrita en formato de caracteres. La salida es una serie de caracteres, como por ejemplo 2010/04/23.

CALL CEEDATE, sintaxis



***fecha_Lilian_entrada* (entrada)**
Un entero de 32 bits que representa la fecha de Lilian. La fecha Liliana es el número de días desde el 14 de octubre de 1582. Por ejemplo, el 16 de mayo de 1988 es el día de Lilian número 148138. El rango válido de fechas Lilian es de 1 a 3.074.324 (15 de octubre de 1582 a 31 de diciembre de 9999).

***picture_string* (entrada)**
Una serie de caracteres con prefijo de longitud de media palabra que representa el formato necesario de *output_char_date*, por ejemplo MM/DD/YY. Cada carácter de *picture_string* representa un carácter de *output_char_date*. Si los delimitadores como la barra inclinada (/) aparecen en la serie de imagen, se copian tal cual en *output_char_date*.

Si *picture_string* incluye un símbolo de era japonesa <JJJJ>, la posición YY en *output_char_date* se sustituye por el número de año dentro de la era japonesa. Por ejemplo, el año 1988 equivale al año japonés 63 en la era Showa.

***output_char_date* (salida)**
Una serie de 80 caracteres de longitud fija que es el resultado de convertir *input_Lilian_date* al formato especificado por *picture_string*. Si *input_Lilian_date* no es válido, *output_char_date* se establece en todos los espacios en blanco y CEEDATE termina con un código de comentarios simbólico non-CEE000 .

***fc* (salida)**
Un código de retroalimentación de 12 bytes (opcional) que indica el resultado de este servicio.

Tabla 57. ***Condiciones simbólicas de CEEDATE***			
Código de comentarios simbólicos	**Gravedad**	**Número de mensaje**	**texto de mensaje**
CEE000	0	--	El servicio se ha completado correctamente.
CEE2EG	3	2512	El valor de fecha Lilian pasado en una llamada a CEEDATE o CEEDYWK no estaba dentro del rango soportado.
CEE2EM	3	2518	Se ha especificado una serie de imagen no válida en una llamada a un servicio de fecha/hora.
CEE2EQ	3	2522	Se ha utilizado una era (< JJJJ>, < CCCC> o < CCCCCCCCCC>) en una serie de imagen pasada a CEEDATE, pero el valor de fecha de Lilian no estaba dentro del rango soportado. No se pudo determinar la era.
CEE2EU	2	2526	La serie de fecha devuelta por CEEDATE se ha truncado.
CEE2F6	1	2534	Se ha especificado una anchura de campo insuficiente para un nombre de mes o día de la semana en una llamada a CEEDATE o CEEDATM. La salida se ha establecido en blancos.

Nota de uso: El inverso de CEEDATE es CEEDAYS, que convierte las fechas de caracteres al formato Lilian.

ejemplo

```
*************************************************
**                                             **
** Function: CEEDATE - convert Lilian date to **
**                     character format        **
**                                             **
** In this example, a call is made to CEEDATE **
** to convert a Lilian date (the number of    **
** days since 14 October 1582) to a character **
** format (such as 6/22/98). The result is    **
** displayed.  The Lilian date is obtained    **
** via a call to CEEDAYS.                      **
**                                             **
*************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. CBLDATE.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  LILIAN                  PIC S9(9) BINARY.
 01  CHRDATE                 PIC X(80).
 01  IN-DATE.
     02  Vstring-length      PIC S9(4) BINARY.
     02  Vstring-text.
         03  Vstring-char    PIC X
                     OCCURS 0 TO 256 TIMES
                     DEPENDING ON Vstring-length
                         of IN-DATE.
 01  PICSTR.
     02  Vstring-length      PIC S9(4) BINARY.
     02  Vstring-text.
         03  Vstring-char    PIC X
                     OCCURS 0 TO 256 TIMES
                     DEPENDING ON Vstring-length
                        of PICSTR.
 01  FC.
     02  Condition-Token-Value.
     COPY  CEEIGZCT.
         03  Case-1-Condition-ID.
             04  Severity    PIC S9(4) COMP.
             04  Msg-No      PIC S9(4) COMP.
         03  Case-2-Condition-ID
                REDEFINES Case-1-Condition-ID.
             04  Class-Code  PIC S9(4) COMP.
```

```
             04  Cause-Code  PIC S9(4) COMP.
         03  Case-Sev-Ctl    PIC X.
         03  Facility-ID     PIC XXX.
     02  I-S-Info            PIC S9(9) COMP.
*
 PROCEDURE DIVISION.
 PARA-CBLDAYS.
*********************************************
** Call CEEDAYS to convert date of 6/2/98 to   **
**     Lilian representation                   **
*********************************************
     MOVE 6 TO Vstring-length of IN-DATE.
     MOVE '6/2/98' TO Vstring-text of IN-DATE(1:6).
     MOVE 8 TO Vstring-length of PICSTR.
     MOVE 'MM/DD/YY' TO Vstring-text of PICSTR(1:8).
     CALL 'CEEDAYS' USING IN-DATE, PICSTR,
                          LILIAN, FC.

*********************************************
** If CEEDAYS runs successfully, display result**
*********************************************
     IF  CEE000 of FC  THEN
         DISPLAY Vstring-text of IN-DATE
             ' is Lilian day: ' LILIAN
     ELSE
         DISPLAY 'CEEDAYS failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

*********************************************
** Specify picture string that describes the   **
**  required format of the output from CEEDATE, **
**  and the picture string's length.           **
*********************************************
     MOVE 23 TO Vstring-length OF PICSTR.
     MOVE 'ZD Mmmmmmmmmmmmmmz YYYY' TO
                  Vstring-text OF PICSTR(1:23).

*********************************************
** Call CEEDATE to convert the Lilian date     **
**     to  a picture string.                   **
*********************************************
     CALL 'CEEDATE' USING LILIAN, PICSTR,
                          CHRDATE, FC.

*********************************************
** If CEEDATE runs successfully, display result**
*********************************************
     IF CEE000 of FC  THEN
         DISPLAY 'Input Lilian date of ' LILIAN
             ' corresponds to:  ' CHRDATE
     ELSE
         DISPLAY 'CEEDATE failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

     GOBACK.
```

La tabla siguiente muestra la salida de ejemplo de CEEDATE.

fecha_Lilian_entrada	serie_ilustración	fecha_char_salida
148138	YY	98
	YYMM	9805
	YY-MM	98-05
	YYMMDD	980516
	YYYYMMDD	19980516
	YYYY-MM-DD	1998-05-16
	YYYY-ZM-ZD	1998-5-16
	<JJJJ> YY.MM.DD	Showa 63.05.16 (en una serie DBCS)
148139	MM	05
	MMDD	0517
	MM/DD	05/17
	MMDDYY	051798
	MM/DD/YYYY	05/17/1998
	ZM/DD/YYYY	5/17/1998
148140	DD	18
	DDMM	1805
	DDMMYY	180598
	DD.MM.YY	18.05.98
	DD.MM.YYYY	18.05.1998
	DD Mmm YYYY	18 mayo 1998
148141	DDD	140
	YYDDD	98140
	YY.DDD	98.140
	YYYY.DDD	1998.140
148142	YY/MM/DD HH:MI:SS.99	98/05/20 00:00:00.00
	YYYY/ZM/ZD ZH:MI AP	1998/5/20 0:00 AM
148143	WWW., MMM DD, YYYY	SAT., MAYO 21, 1998
	Www., Mmm DD, YYYY	Sat., 21 de mayo de 1998
	Wwwwwwwwww, Mmmmmmmmmm DD, YYYY	Sábado, 21 de mayo de 1998
	Wwwwwwwwwz, Mmmmmmmmmz DD, YYYY	Sábado, 21 de mayo de 1998

“Ejemplo: series de imágenes de fecha y hora” en la página 545

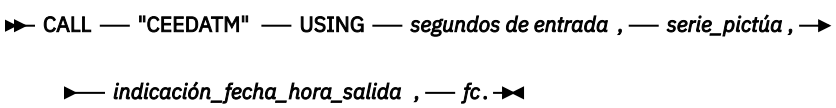
Referencias relacionadas

“Términos y series de caracteres de imagen” en la página 542

CEEDATM: convertir segundos en indicación de fecha y hora de tipo carácter

CEEDATM convierte un número que representa el número de segundos desde 00:00:00 14 de octubre de 1582 en una serie de caracteres. La salida es una indicación de fecha y hora de serie de caracteres como, por ejemplo, `1988/07/26 20:37:00`.

CALL CEEDATM, sintaxis



***input_seconds* (entrada)**
Un número de coma flotante de 64 bits que representa el número de segundos desde las 00:00:00 del 14 de octubre de 1582, sin contar los segundos bisiestos.

Por ejemplo, 00:00:01 el 15 de octubre de 1582 es el segundo número 86,401 (24 * 60 * 60 + 01). El rango válido de *input_seconds* es de 86,400 a 265.621.679.999,999 (23:59:59.999 31 de diciembre 9999).

***picture_string* (entrada)**
Una serie de caracteres con prefijo de longitud de media palabra que representa el formato necesario de *output_timestamp*, por ejemplo, `MM/DD/YY HH:MI AP`.

Cada carácter de *picture_string* representa un carácter de *output_timestamp*. Si se utilizan delimitadores como una barra inclinada (/) en la serie de imagen, se copian tal cual en *output_timestamp*.

Si *picture_string* incluye el símbolo de la era japonesa <JJJJ>, la posición YY en *output_timestamp* representa el año dentro de la era japonesa.

***output_timestamp* (salida)**
Una serie de 80 caracteres de longitud fija que es el resultado de convertir *input_seconds* al formato especificado por *picture_string*.

Si es necesario, la salida se trunca a la longitud de *output_timestamp*.

Si *input_seconds* no es válido, *output_timestamp* se establece en todos los espacios en blanco y CEEDATM termina con un código de retorno simbólico non-CEE000 .

***fc* (salida)**
Un código de retroalimentación de 12 bytes (opcional) que indica el resultado de este servicio.

Tabla 58. ***Condiciones simbólicas de CEEDATM***

Código de comentarios simbólicos	Gravedad	Número de mensaje	texto de mensaje
CEE000	0	--	El servicio se ha completado correctamente.
CEE2E9	3	2505	El valor de input_seconds en una llamada a CEEDATM o CEESECI no estaba dentro del rango soportado.
CEE2EA	3	2506	Se ha utilizado una era (< JJJ>, < CCCC> o < CCCCCCCCCC>) en una serie de imagen pasada a CEEDATM, pero el valor de número de segundos de entrada no estaba dentro del rango soportado. No se pudo determinar la era.
CEE2EM	3	2518	Se ha especificado una serie de imagen no válida en una llamada a un servicio de fecha u hora.

*Tabla 58. **Condiciones simbólicas de CEEDATM** (continuación)*			
Código de comentarios simbólicos	**Gravedad**	**Número de mensaje**	**texto de mensaje**
CEE2EV	2	2527	La serie de indicación de fecha y hora devuelta por CEEDATM se ha truncado.
CEE2F6	1	2534	Se ha especificado una anchura de campo insuficiente para un nombre de mes o día de la semana en una llamada a CEEDATE o CEEDATM. La salida se ha establecido en blancos.

Nota de uso: El inverso de CEEDATM es CEESECS, que convierte una indicación de fecha y hora en número de segundos.

ejemplo

```
************************************************
**                                            **
** Function: CEEDATM - convert seconds to     **
**                     character time stamp   **
**                                            **
** In this example, a call is made to CEEDATM  **
** to convert a date represented in Lilian    **
** seconds (the number of seconds since       **
** 00:00:00 14 October 1582) to a character   **
** format (such as 06/02/88 10:23:45). The    **
** result is displayed.                       **
**                                            **
************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. CBLDATM.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  DEST          PIC S9(9) BINARY VALUE 2.
 01  SECONDS                 COMP-2.
 01  IN-DATE.
     02  Vstring-length      PIC S9(4) BINARY.
     02  Vstring-text.
         03  Vstring-char    PIC X
                     OCCURS 0 TO 256 TIMES
                     DEPENDING ON Vstring-length
                         of IN-DATE.
 01  PICSTR.
     02  Vstring-length      PIC S9(4) BINARY.
     02  Vstring-text.
         03  Vstring-char    PIC X
                     OCCURS 0 TO 256 TIMES
                     DEPENDING ON Vstring-length
                        of PICSTR.
 01  TIMESTP                 PIC X(80).
 01  FC.
     02  Condition-Token-Value.
     COPY  CEEIGZCT.
         03  Case-1-Condition-ID.
             04  Severity    PIC S9(4) COMP.
             04  Msg-No      PIC S9(4) COMP.
         03  Case-2-Condition-ID
                    REDEFINES Case-1-Condition-ID.
             04  Class-Code  PIC S9(4) COMP.
             04  Cause-Code  PIC S9(4) COMP.
         03  Case-Sev-Ctl    PIC X.
         03  Facility-ID     PIC XXX.
     02  I-S-Info            PIC S9(9) COMP.
*
 PROCEDURE DIVISION.
 PARA-CBLDATM.
************************************************
** Call CEESECS to convert time stamp of 6/2/88**
**     at 10:23:45 AM to Lilian representation **
************************************************
     MOVE 20 TO Vstring-length of IN-DATE.
     MOVE '06/02/88 10:23:45 AM'
```

```
         TO Vstring-text of IN-DATE.
     MOVE 20 TO Vstring-length of PICSTR.
     MOVE 'MM/DD/YY HH:MI:SS AP'
         TO Vstring-text of PICSTR.
     CALL 'CEESECS' USING IN-DATE, PICSTR,
                          SECONDS, FC.

************************************************
** If CEESECS runs successfully, display result**
************************************************
     IF  CEE000 of FC  THEN
         DISPLAY Vstring-text of IN-DATE
             ' is Lilian second:  ' SECONDS
     ELSE
         DISPLAY 'CEESECS failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

************************************************
** Specify required format of the output.       **
************************************************
     MOVE 35 TO Vstring-length OF PICSTR.
     MOVE 'ZD Mmmmmmmmmmmmmmz YYYY at HH:MI:SS'
             TO Vstring-text OF PICSTR.

************************************************
** Call CEEDATM to convert Lilian seconds to   **
**     a character time stamp                   **
************************************************
     CALL 'CEEDATM' USING SECONDS, PICSTR,
                          TIMESTP, FC.

************************************************
** If CEEDATM runs successfully, display result**
************************************************
     IF CEE000 of FC  THEN
         DISPLAY 'Input seconds of ' SECONDS
             ' corresponds to: ' TIMESTP
     ELSE
         DISPLAY 'CEEDATM failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

     GOBACK.
```

Las tablas siguientes muestran una salida de ejemplo de CEEDATM.

segundos de entrada	**serie_ilustración**	**fecha_fecha_hora_salida**
12,799,191,601.000	YYMMDD	880516
	HH:MI:SS	19:00:01
	YY-MM-DD	88-05-16
	YYMMDDHHMISS	880516190001
	YY-MM-DD HH:MI:SS	88-05-16 19:00:01
	YYYY-MM-DD HH:MI:SS AP	1988-05-16 07:00:01 PM
12,799,191,661.986	DD Mmm YY	16 mayo 88
	DD MMM YY HH:MM	16 DE MAYO 88 19:01
	WWW, MMM DD, YYYY	MON, 16 de mayo de 1988
	ZH:MI AP	7:01 PM
	Wwwwwwwwwz, ZM/ZD/YY	Lunes, 5/16/88
	HH:MI:SS.99	19:01:01.98

segundos de entrada	serie_ilustración	fecha_fecha_hora_salida
12,799,191,662.009	YYYY	1988
	YY	88
	Y	8
	MM	05
	ZM	5
	RRRR	V

segundos de entrada	serie_ilustración	fecha_fecha_hora_salida
12,799,191,662.009	MMM	MAY
	Mmm	May
	Mmmmmmmmm	May
	Mmmmmmmmmz	May
	DD	16
	ZD	16
	DDD	137
	HH	19
	ZH	19
	MI	01
	SS	02
	99	00
	999	009
	AP	PM
	WWW	MON
	Www	Mon
	Wwwwwwwww	Monday
	Wwwwwwwwz	Monday

“Ejemplo: series de imágenes de fecha y hora” en la página 545

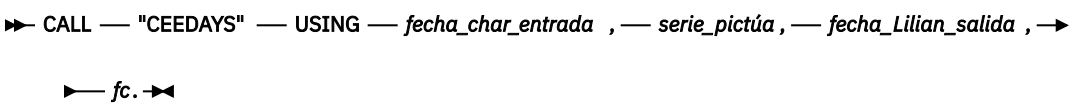
Referencias relacionadas
“Términos y series de caracteres de imagen” en la página 542

CEEDAYS: convertir fecha a formato Lilian

CEEDAYS convierte una serie que representa una fecha en un formato Lilian, que representa una fecha como el número de días desde el principio del calendario gregoriano (viernes, 14 de octubre, 1582).

No utilice CEEDAYS en combinación con funciones intrínsecas COBOL. Utilice CEECBLDY para programas que utilizan funciones intrínsecas.

Sintaxis de CALL CEEDAYS



***fecha_char_entrada* (entrada)**

Una serie de caracteres con prefijo de longitud de media palabra que representa una fecha o una indicación de fecha y hora en un formato que se ajusta al especificado por *picture_string*.

La serie de caracteres debe contener entre 5 y 255 caracteres, ambos inclusive. *fecha_char_entrada* puede contener espacios en blanco iniciales o finales. El análisis de una fecha empieza por el primer carácter no en blanco (a menos que la propia serie de imagen contenga espacios en blanco iniciales, en cuyo caso CEEDAYS omite exactamente esa cantidad de posiciones antes de que comience el análisis).

Después de analizar una fecha válida, tal como determina el formato de la fecha especificada en *picture_string*, CEEDAYS ignora todos los caracteres restantes. Las fechas válidas están comprendidas entre el 15 de octubre de 1582 y el 31 de diciembre de 9999.

***picture_string* (entrada)**

Una serie de caracteres con prefijo de longitud de media palabra, que indica el formato de la fecha especificada en *fecha_char_entrada*.

Cada carácter de *picture_string* corresponde a un carácter de *input_char_date*. Por ejemplo, si especifica MMDDYY como *picture_string*, CEEDAYS lee una *fecha_char_entrada* de 060288 como 02 de junio de 1988.

Si aparecen delimitadores como una barra inclinada (/) en la serie de imagen, se pueden omitir los ceros iniciales. Por ejemplo, las llamadas siguientes a CEEDAYS asignan cada una el mismo valor, 148155 (02 de junio de 1988), a *lildate*:

```
CALL CEEDAYS USING '6/2/88'  , 'MM/DD/YY', lildate, fc.
CALL CEEDAYS USING '06/02/88', 'MM/DD/YY', lildate, fc.
CALL CEEDAYS USING '060288'  , 'MMDDYY'  , lildate, fc.
CALL CEEDAYS USING '88154'   , 'YYDDD'   , lildate, fc.
```

Siempre que se incluyan caracteres como, por ejemplo, dos puntos o barras inclinadas en la *picture_string* (como `HH:MI:SS YY/MM/DD`), se contabilizarán como marcadores de posición, pero por lo demás se ignorarán.

Si *picture_string* incluye un símbolo de era japonesa `<JJJJ>`, la posición YY en *input_char_date* se sustituye por el número de año dentro de la era japonesa. Por ejemplo, el año 1988 equivale al año japonés 63 en la era Showa.

***output_Lilian_date* (salida)**

Un entero binario de 32 bits que representa la fecha de Lilian, el número de días desde el 14 de octubre de 1582. Por ejemplo, el 16 de mayo de 1988 es el día 148138.

Si *input_char_date* no contiene una fecha válida, *output_Lilian_date* se establece en 0 y CEEDAYS termina con un código de retorno simbólico non-CEE000 .

Los cálculos de fecha se realizan fácilmente en *output_Lilian_date*, porque es un entero. Las anomalías de año bisiesto y de fin de año no afectan a los cálculos.

***fc* (salida)**

Un código de retroalimentación de 12 bytes (opcional) que indica el resultado de este servicio.

Tabla 59. ***Condiciones simbólicas de CEEDAYS***			
Código de comentarios simbólicos	**Gravedad**	**Número de mensaje**	**texto de mensaje**
CEE000	0	--	El servicio se ha completado correctamente.
CEE2EB	3	2507	No se pasaron datos suficientes a CEEDAYS o CEESECS. El valor Lilian no se ha calculado.
CEE2EC	3	2508	El valor de fecha pasado a CEEDAYS o CEESECS no era válido.
CEE2ED	3	2509	La era pasada a CEEDAYS o CEESECS no fue reconocida.
CEE2EH	3	2513	La fecha de entrada pasada en una llamada CEEISEC, CEEDAYS o CEESECS no estaba dentro del rango soportado.
CEE2EL	3	2517	No se ha reconocido el valor de mes en una llamada CEEISEC.
CEE2EM	3	2518	Se ha especificado una serie de imagen no válida en una llamada a un servicio de fecha/hora.
CEE2EO	3	2520	CEEDAYS ha detectado datos no numéricos en un campo numérico, o la serie de fecha no coincidía con la serie de imagen.
CEE2EP	3	2521	El valor de año dentro de era < JJJJ>, < CCCC> o < CCCCCCCCCC> pasado a CEEDAYS o CEESECS era cero.

Notas de uso

- El inverso de CEEDAYS es CEEDATE, que convierte *output_Lilian_date* del formato Lilian al formato de caracteres.
- Para realizar cálculos en fechas anteriores al 15 de octubre de 1582, añada 4000 al año en cada fecha, convierta las fechas a Lilian y, a continuación, realice el cálculo. Si el resultado del cálculo es una fecha, en contraposición a un número de días, convierta el resultado en una serie de fecha y reste 4000 del año.
- De forma predeterminada, los años de dos dígitos se encuentran dentro del rango de 100 años que empieza 80 años antes de la fecha del sistema. Así en 2010, todos los años de dos dígitos representan fechas entre 1930 y 2029, inclusive. Puede cambiar el rango predeterminado utilizando el servicio invocable CEESCEN.
- Puede realizar fácilmente cálculos de fecha en *output_Lilian_date*, porque es un entero. Se evitan anomalías de año bisiesto y de fin de año.

ejemplo

```
*****************************************
**                                     **
** Function: CEEDAYS - convert date to   **
**                     Lilian format     **
**                                     **
*****************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. CBLDAYS.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  CHRDATE.
     02  Vstring-length      PIC S9(4) BINARY.
     02  Vstring-text.
         03  Vstring-char    PIC X
                     OCCURS 0 TO 256 TIMES
                     DEPENDING ON Vstring-length
                         of CHRDATE.
```

```
 01  PICSTR.
     02  Vstring-length      PIC S9(4) BINARY.
     02  Vstring-text.
         03  Vstring-char    PIC X
                     OCCURS 0 TO 256 TIMES
                     DEPENDING ON Vstring-length
                         of PICSTR.
 01  LILIAN                  PIC S9(9) BINARY.
 01  FC.
     02  Condition-Token-Value.
     COPY  CEEIGZCT.
         03  Case-1-Condition-ID.
             04  Severity    PIC S9(4) COMP.
             04  Msg-No      PIC S9(4) COMP.
         03  Case-2-Condition-ID
                   REDEFINES Case-1-Condition-ID.
             04  Class-Code  PIC S9(4) COMP.
             04  Cause-Code  PIC S9(4) COMP.
         03  Case-Sev-Ctl    PIC X.
         03  Facility-ID     PIC XXX.
     02  I-S-Info            PIC S9(9) COMP.
*
 PROCEDURE DIVISION.
 PARA-CBLDAYS.
************************************************
** Specify input date and length               **
************************************************
     MOVE 16 TO Vstring-length of CHRDATE.
     MOVE '1 January 2005'
         TO Vstring-text of CHRDATE.

************************************************
** Specify a picture string that describes     **
** input date, and the picture string's length.**
************************************************
     MOVE 25 TO Vstring-length of PICSTR.
     MOVE 'ZD Mmmmmmmmmmmmmmz YYYY'
             TO Vstring-text of PICSTR.

************************************************
** Call CEEDAYS to convert input date to a     **
** Lilian date                                 **
************************************************
     CALL 'CEEDAYS' USING CHRDATE, PICSTR,
                          LILIAN, FC.

************************************************
** If CEEDAYS runs successfully, display result**
************************************************
     IF  CEE000 of FC  THEN
         DISPLAY Vstring-text of CHRDATE
             ' is Lilian day: ' LILIAN
     ELSE
         DISPLAY 'CEEDAYS failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

     GOBACK.
```

"Ejemplo: series de imágenes de fecha y hora" en la página 545

Referencias relacionadas

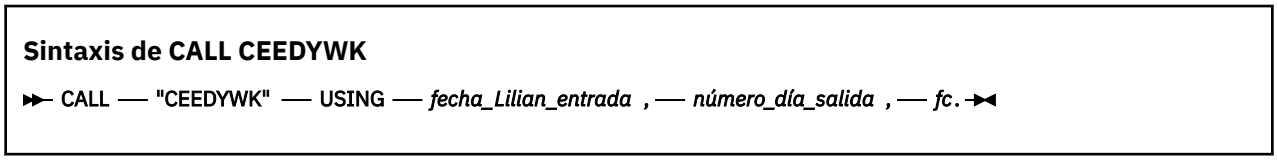
"Términos y series de caracteres de imagen" en la página 542

CEEDYWK: calcular día de la semana a partir de la fecha lililiana

CEEDYWK calcula el día de la semana en el que una fecha Lilian cae como un número entre 1 y 7.

El número devuelto por CEEDYWK es útil para los cálculos de fin de semana.

Sintaxis de CALL CEEDYWK



***fecha_Lilian_entrada* (entrada)**
Un entero binario de 32 bits que representa la fecha de Lilian, el número de días desde el 14 de octubre de 1582.

Por ejemplo, el 16 de mayo de 1988 es el día 148138. El rango válido de *input_Lilian_date* está entre 1 y 3.074.324 (15 de octubre de 1582 y 31 de diciembre de 9999).

***output_day_no* (salida)**
Un entero binario de 32 bits que representa el día de la semana de *input_Lilian_date*: 1 es igual al domingo, 2 es igual al lunes. ., 7 es igual al sábado.

Si *input_Lilian_date* no es válido, *output_day_no* se establece en 0 y CEEDYWK termina con un código de comentarios simbólico non-CEE000 .

***fc* (salida)**
Un código de retroalimentación de 12 bytes (opcional) que indica el resultado de este servicio.

Tabla 60. ***Condiciones simbólicas CEEDYWK***

Código de comentarios simbólicos	Gravedad	Número de mensaje	texto de mensaje
CEE000	0	--	El servicio se ha completado correctamente.
CEE2EG	3	2512	El valor de fecha Lilian pasado en una llamada a CEEDATE o CEEDYWK no estaba dentro del rango soportado.

ejemplo

```
************************************************
**                                            **
** Function: Call CEEDYWK to calculate the    **
**           day of the week from Lilian date **
**                                            **
** In this example, a call is made to CEEDYWK **
** to return the day of the week on which a   **
** Lilian date falls. (A Lilian date is the   **
** number of days since 14 October 1582)      **
**                                            **
************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. CBLDYWK.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  LILIAN                  PIC S9(9) BINARY.
 01  DAYNUM                  PIC S9(9) BINARY.
 01  IN-DATE.
     02  Vstring-length      PIC S9(4) BINARY.
     02  Vstring-text.
         03  Vstring-char        PIC X,
                     OCCURS 0 TO 256 TIMES
                     DEPENDING ON Vstring-length
                         of IN-DATE.
 01  PICSTR.
     02  Vstring-length      PIC S9(4) BINARY.
     02  Vstring-text.
         03  Vstring-char        PIC X,
                     OCCURS 0 TO 256 TIMES
                     DEPENDING ON Vstring-length
                         of PICSTR.
 01  FC.
     02  Condition-Token-Value.
     COPY  CEEIGZCT.
         03  Case-1-Condition-ID.
             04  Severity    PIC S9(4) COMP.
             04  Msg-No      PIC S9(4) COMP.
         03  Case-2-Condition-ID
                   REDEFINES Case-1-Condition-ID.
             04  Class-Code  PIC S9(4) COMP.
             04  Cause-Code  PIC S9(4) COMP.
         03  Case-Sev-Ctl    PIC X.
```

```
         03  Facility-ID     PIC XXX.
     02  I-S-Info            PIC S9(9) COMP.

 PROCEDURE DIVISION.
 PARA-CBLDAYS.
** Call CEEDAYS to convert date of 6/2/88 to
**     Lilian representation
     MOVE 6 TO Vstring-length of IN-DATE.
     MOVE '6/2/88' TO Vstring-text of IN-DATE(1:6).
     MOVE 8 TO Vstring-length of PICSTR.
     MOVE 'MM/DD/YY' TO Vstring-text of PICSTR(1:8).
     CALL 'CEEDAYS' USING IN-DATE, PICSTR,
         LILIAN, FC.

** If CEEDAYS runs successfully, display result.
     IF  CEE000 of FC  THEN
         DISPLAY Vstring-text of IN-DATE
             ' is Lilian day: ' LILIAN
     ELSE
         DISPLAY 'CEEDAYS failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

 PARA-CBLDYWK.

** Call CEEDYWK to return the day of the week on
** which the Lilian date falls
     CALL 'CEEDYWK' USING LILIAN , DAYNUM , FC.

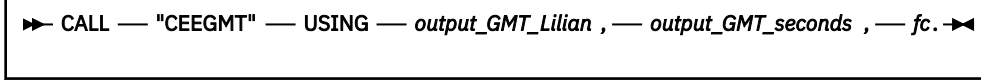
** If CEEDYWK runs successfully, print results
     IF CEE000 of FC  THEN
         DISPLAY 'Lilian day ' LILIAN
             ' falls on day ' DAYNUM
             ' of the week, which is a:'
** Select DAYNUM to display the name of the day
**     of the week.
         EVALUATE DAYNUM
           WHEN 1
             DISPLAY 'Sunday.'
           WHEN 2
             DISPLAY 'Monday.'
           WHEN 3
             DISPLAY 'Tuesday'
           WHEN 4
             DISPLAY 'Wednesday.'
           WHEN 5
             DISPLAY 'Thursday.'
           WHEN 6
             DISPLAY 'Friday.'
           WHEN 7
             DISPLAY 'Saturday.'
         END-EVALUATE
     ELSE
         DISPLAY 'CEEDYWK failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

     GOBACK.
```

CEEGMT: obtener hora media de Greenwich actual

CEEGMT devuelve la hora media de Greenwich actual (GMT) como fecha de Lilian y como número de segundos desde 00:00:00 del 14 de octubre de 1582. Los valores devueltos son compatibles con los generados y utilizados por los otros servicios invocables de Fecha y hora .

Sintaxis de CALL CEEGMT



***output_GMT_Lilian* (salida)**

Un entero binario de 32 bits que representa la fecha actual en Greenwich, Inglaterra, en el formato Lilian (el número de días desde el 14 de octubre de 1582).

Por ejemplo, el 16 de mayo de 1988 es el día 148138. Si GMT no está disponible en el sistema, *output_GMT_Lilian* se establece en 0 y CEEGMT termina con un código de retorno simbólico non-CEE000 .

***output_GMT_seconds* (salida)**

Un número de coma flotante largo de 64 bits que representa la fecha y hora actual en Greenwich, Inglaterra, como el número de segundos desde 00:00:00 el 14 de octubre de 1582, sin contar los segundos bisiestos.

Por ejemplo, 00:00:01 el 15 de octubre de 1582 es el segundo número 86,401 (24 * 60 * 60 + 01). 19 :00:01.078 El 16 de mayo de 1988 es el segundo número 12,799,191,601.078. Si GMT no está disponible en el sistema, *output_GMT_seconds* se establece en 0 y CEEGMT termina con un código de comentarios simbólico non-CEE000 .

***fc* (salida)**

Un código de retroalimentación de 12 bytes (opcional) que indica el resultado de este servicio.

Tabla 61. ***Condiciones simbólicas CEEGMT***

Código de comentarios simbólicos	Gravedad	Número de mensaje	texto de mensaje
CEE000	0	--	El servicio se ha completado correctamente.
CEE2E6	3	2502	El UTC/GMT no estaba disponible en el sistema.

Notas de uso

- CEEDATE convierte *output_GMT_Lilian* en una fecha de tipo carácter y CEEDATM convierte *output_GMT_seconds* en una indicación de fecha y hora de tipo carácter.
- Para que los resultados de este servicio sean significativos, el reloj del sistema debe establecerse en la hora local y la variable de entorno TZ debe establecerse correctamente.
- Los valores devueltos por CEEGMT son útiles para los cálculos de tiempo transcurrido. Por ejemplo, puede calcular el tiempo transcurrido entre dos llamadas a CEEGMT calculando las diferencias entre los valores devueltos.
- CEEUTC es idéntico a este servicio.

ejemplo

```
************************************************
**                                            **
** Function: Call CEEGMT to get current       **
**           Greenwich Mean Time              **
**                                            **
** In this example, a call is made to CEEGMT   **
** to return the current GMT as a Lilian date  **
** and as Lilian seconds. The results are      **
** displayed.                                  **
**                                            **
************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. IGZTGMT.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  LILIAN                  PIC S9(9) BINARY.
 01  SECS                    COMP-2.
 01  FC.
     02  Condition-Token-Value.
     COPY  CEEIGZCT.
         03  Case-1-Condition-ID.
             04  Severity    PIC S9(4) COMP.
             04  Msg-No      PIC S9(4) COMP.
         03  Case-2-Condition-ID
                   REDEFINES Case-1-Condition-ID.
             04  Class-Code  PIC S9(4) COMP.
             04  Cause-Code  PIC S9(4) COMP.
```

```
         03  Case-Sev-Ctl    PIC X.
         03  Facility-ID     PIC XXX.
     02  I-S-Info            PIC S9(9) COMP.
PROCEDURE DIVISION.
PARA-CBLGMT.
     CALL 'CEEGMT' USING LILIAN , SECS , FC.

     IF CEE000 of FC  THEN
         DISPLAY 'The current GMT is also '
             'known as Lilian day: ' LILIAN
         DISPLAY 'The current GMT in Lilian '
             'seconds is: ' SECS
     ELSE
         DISPLAY 'CEEGMT failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

     GOBACK.
```

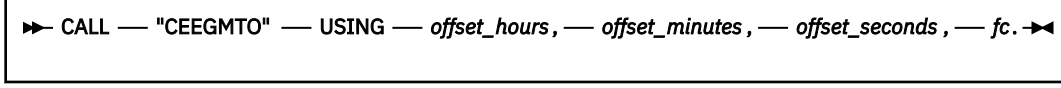
Tareas relacionadas

“Establecimiento de variables de entorno” en la página 229

CEEGMTO: obtener desplazamiento de la hora media de Greenwich a la hora local

CEEGMTO devuelve valores a la rutina de llamada que representan la diferencia entre la hora del sistema local y la hora media de Greenwich (GMT).

Sintaxis de CALL CEEGMTO



***offset_hours* (salida)**

Un entero binario de 32 bits que representa el desplazamiento de GMT a la hora local, en horas.

Por ejemplo, para la hora estándar del Pacífico, *offset_hours* es igual a -8.

El rango de *offset_hours* es de -12 a +13 (+ 13 = horario de verano en el huso horario +12).

Si el desplazamiento de hora local no está disponible, *offset_hours* es igual a 0 y CEEGMTO termina con un código de comentarios simbólico non-CEE000 .

***offset_minutes* (salida)**

Un entero binario de 32 bits que representa el número de minutos adicionales que la hora local está por delante o por detrás de GMT.

El rango de *offset_minutes* es de 0 a 59.

Si el desplazamiento de hora local no está disponible, *offset_minutes* es igual a 0 y CEEGMTO termina con un código de comentarios simbólico non-CEE000 .

***offset_seconds* (salida)**

Número de coma flotante largo de 64 bits que representa el desplazamiento de GMT a la hora local, en segundos.

Por ejemplo, la hora estándar del Pacífico está ocho horas por detrás de GMT. Si la hora local está en el huso horario del Pacífico durante la hora estándar, CEEGMTO devolvería -28,800 (-8 * 60 * 60). El rango de *offset_seconds* es de -43.200 a +46.800. *offset_seconds* se puede utilizar con CEEGMT para calcular la fecha y hora local.

Si el desplazamiento de hora local no está disponible desde el sistema, *offset_seconds* se establece en 0 y CEEGMTO termina con un código de retorno simbólico non-CEE000 .

***fc* (salida)**

Un código de retroalimentación de 12 bytes (opcional) que indica el resultado de este servicio.

Tabla 62. ***Condiciones simbólicas de CEEGMTO***			
Código de comentarios simbólicos	**Gravedad**	**Número de mensaje**	**texto de mensaje**
CEE000	0	--	El servicio se ha completado correctamente.
CEE2E7	3	2503	El desplazamiento de UTC/GMT a la hora local no estaba disponible desde el sistema.

Notas de uso

- CEEDATM convierte *offset_seconds* en una indicación de fecha y hora de tipo carácter.
- Para que los resultados de este servicio sean significativos, el reloj del sistema debe establecerse en la hora local y la variable de entorno TZ debe establecerse correctamente.

ejemplo

```
************************************************
**                                            **
** Function:  Call CEEGMTO to get offset from  **
**            Greenwich Mean Time to local     **
**            time                             **
**                                            **
** In this example, a call is made to CEEGMTO  **
** to return the offset from GMT to local time **
** as separate binary integers representing    **
** offset hours, minutes, and seconds. The     **
** results are displayed.                      **
**                                            **
************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. IGZTGMTO.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  HOURS                   PIC S9(9) BINARY.
 01  MINUTES                 PIC S9(9) BINARY.
 01  SECONDS COMP-2.
 01  FC.
     02  Condition-Token-Value.
     COPY  CEEIGZCT.
         03  Case-1-Condition-ID.
             04  Severity    PIC S9(4) COMP.
             04  Msg-No      PIC S9(4) COMP.
         03  Case-2-Condition-ID
                    REDEFINES Case-1-Condition-ID.
             04  Class-Code  PIC S9(4) COMP.
             04  Cause-Code  PIC S9(4) COMP.
         03  Case-Sev-Ctl    PIC X.
         03  Facility-ID     PIC XXX.
     02  I-S-Info            PIC S9(9) COMP.
 PROCEDURE DIVISION.
 PARA-CBLGMTO.
     CALL 'CEEGMTO' USING HOURS , MINUTES ,
         SECONDS , FC.

     IF CEE000 of FC  THEN
         DISPLAY 'Local time differs from GMT '
             'by: ' HOURS ' hours, '
             MINUTES ' minutes, OR  '
             SECONDS ' seconds. '
     ELSE
         DISPLAY 'CEEGMTO failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

     GOBACK.
```

Tareas relacionadas

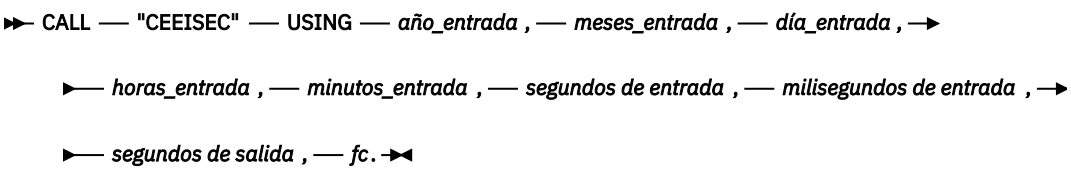
"Establecimiento de variables de entorno" en la página 229

Referencias relacionadas
“Variables de entorno de compilador y tiempo de ejecución” en la página 230
“CEEGMT: obtener hora media de Greenwich actual” en la página 578

CEEISEC: convertir enteros en segundos

CEEISEC convierte enteros binarios que representan año, mes, día, hora, minuto, segundo y milisegundo en un número que representa el número de segundos desde las 00:00:00 del 14 de octubre de 1582.

Sintaxis de CALL CEEISEC



***input_year* (entrada)**
Un entero binario de 32 bits que representa el año.

El rango de valores válidos para *input_year* es de 1582 a 9999, ambos inclusive.

***input_month* (entrada)**
Un entero binario de 32 bits que representa el mes.

El rango de valores válidos para *input_month* es de 1 a 12.

***input_day* (entrada)**
Un entero binario de 32 bits que representa el día.

El rango de valores válidos para *día_entrada* es de 1 a 31.

***input_hours* (entrada)**
Un entero binario de 32 bits que representa las horas.

El rango de valores válidos para *input_hours* es de 0 a 23.

***input_minutes* (entrada)**
Un entero binario de 32 bits que representa los minutos.

El rango de valores válidos para *input_minutes* es de 0 a 59.

***input_seconds* (entrada)**
Un entero binario de 32 bits que representa los segundos.

El rango de valores válidos para *input_seconds* es de 0 a 59.

***input_milisegundos* (entrada)**
Entero binario de 32 bits que representa milisegundos.

El rango de valores válidos para *input_milisegundos* es de 0 a 999.

***output_seconds* (salida)**
Un número de coma flotante de 64 bits que representa el número de segundos desde las 00:00:00 del 14 de octubre de 1582, sin contar los segundos bisiestos.

Por ejemplo, 00:00:01 el 15 de octubre de 1582 es el segundo número 86,401 (24 * 60 * 60 + 01). El rango válido de *output_seconds* es de 86,400 a 265,621,679,999.999 (23:59:59.999 31 de diciembre de 9999).

Si algún valor de entrada no es válido, *output_seconds* se establece en cero.

Para convertir *output_seconds* en un número de día de Lilian, divida *output_seconds* por 86,400 (el número de segundos en un día).

***fc* (salida)**

Un código de retroalimentación de 12 bytes (opcional) que indica el resultado de este servicio.

Tabla 63. ***Condiciones simbólicas CEEISEC***

Código de comentarios simbólicos	Gravedad	Número de mensaje	texto de mensaje
CEE000	0	--	El servicio se ha completado correctamente.
CEE2EE	3	2510	No se ha reconocido el valor de horas en una llamada a CEEISEC o CEESECS.
CEE2EF	3	2511	El parámetro de día pasado en una llamada CEEISEC no era válido para el año y el mes especificados.
CEE2EH	3	2513	La fecha de entrada pasada en una llamada CEEISEC, CEEDAYS o CEESECS no estaba dentro del rango soportado.
CEE2EI	3	2514	El valor de año pasado en una llamada CEEISEC no estaba dentro del rango soportado.
CEE2EJ	3	2515	No se ha reconocido el valor de milisegundos en una llamada CEEISEC.
CEE2EK	3	2516	No se ha reconocido el valor de minutos en una llamada CEEISEC.
CEE2EL	3	2517	No se ha reconocido el valor de mes en una llamada CEEISEC.
CEE2EN	3	2519	No se ha reconocido el valor de segundos en una llamada CEEISEC.

Nota de uso: El inverso de CEEISEC es CEESECI, que convierte el número de segundos en entero año, mes, día, hora, minuto, segundo y milisegundo.

ejemplo

```
************************************************
**                                            **
** Function: Call CEEISEC to convert integers  **
**           to seconds                       **
**                                            **
************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. CBLISEC.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  YEAR                    PIC S9(9) BINARY.
 01  MONTH                   PIC S9(9) BINARY.
 01  DAYS                    PIC S9(9) BINARY.
 01  HOURS                   PIC S9(9) BINARY.
 01  MINUTES                 PIC S9(9) BINARY.
 01  SECONDS                 PIC S9(9) BINARY.
 01  MILLSEC                 PIC S9(9) BINARY.
 01  OUTSECS                 COMP-2.
 01  FC.
     02  Condition-Token-Value.
     COPY  CEEIGZCT.
         03  Case-1-Condition-ID.
             04  Severity    PIC S9(4) COMP.
             04  Msg-No      PIC S9(4) COMP.
         03  Case-2-Condition-ID
                   REDEFINES Case-1-Condition-ID.
             04  Class-Code  PIC S9(4) COMP.
             04  Cause-Code  PIC S9(4) COMP.
         03  Case-Sev-Ctl    PIC X.
         03  Facility-ID     PIC XXX.
```

```
     02  I-S-Info            PIC S9(9) COMP.
 PROCEDURE DIVISION.
 PARA-CBLISEC.
***********************************************
** Specify seven binary integers representing  **
** the date and time as input to be converted  **
** to Lilian seconds                           **
***********************************************
     MOVE 2000 TO YEAR.
     MOVE 1 TO MONTH.
     MOVE 1 TO DAYS.
     MOVE 0 TO HOURS.
     MOVE 0 TO MINUTES.
     MOVE 0 TO SECONDS.
     MOVE 0 TO MILLSEC.
***********************************************
** Call CEEISEC to convert the integers        **
** to seconds                                  **
***********************************************
     CALL 'CEEISEC' USING YEAR, MONTH, DAYS,
                          HOURS, MINUTES, SECONDS,
                          MILLSEC, OUTSECS , FC.
***********************************************
** If CEEISEC runs successfully, display result**
***********************************************
     IF CEE000 of FC  THEN
         DISPLAY MONTH '/' DAYS '/' YEAR
           ' AT ' HOURS ':' MINUTES ':' SECONDS
           ' is equivalent to ' OUTSECS ' seconds'
     ELSE
         DISPLAY 'CEEISEC failed with msg '
           Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

     GOBACK.
```

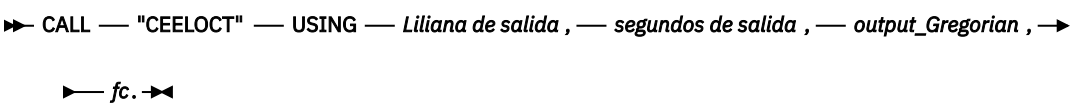
CEELOCT: obtener fecha u hora local actual

CEELOCT devuelve la fecha u hora local actual como una fecha liliana (el número de días desde el 14 de octubre de 1582), como segundos lilianos (el número de segundos desde las 00:00:00 del 14 de octubre de 1582) y como una serie de caracteres gregorianos (YYYYMMDDHHMISS999).

Estos valores son compatibles con otros servicios invocables de Fecha y hora y con las funciones intrínsecas existentes.

CEELOCT realiza la misma función que llamar a los servicios CEEGMT, CEEGMTO y CEEDATM por separado. Llamar a CEELOCT, sin embargo, es mucho más rápido.

Sintaxis de CALL CEELOCT



***output_Lilian* (salida)**

Un entero binario de 32 bits que representa la fecha local actual en el formato Lilian, es decir, el día 1 es igual al 15 de octubre de 1582, el día 148.887 es igual al 4 de junio de 1990.

Si la hora local no está disponible en el sistema, *output_Lilian* se establece en 0 y CEELOCT termina con un código de retorno simbólico non-CEE000 .

***output_seconds* (salida)**

Un número de coma flotante largo de 64 bits que representa la fecha y hora local actual como el número de segundos desde 00:00:00 el 14 de octubre de 1582, sin contar los segundos bisiestos. Por ejemplo, 00:00:01 el 15 de octubre de 1582 es el segundo número 86,401 (24 * 60 * 60 + 01). 19 :00:01.078 el 4 de junio de 1990 es el segundo número 12,863,905,201.078.

Si la hora local no está disponible desde el sistema, *output_seconds* se establece en 0 y CEELOCT termina con un código de retorno simbólico non-CEE000 .

***output_Gregorian* (salida)**

Serie de caracteres de longitud fija de 17 bytes con el formato YYYYMMDDHHMISS999 que representa el año local, el mes, el día, la hora, el minuto, el segundo y el milisegundo.

Si el formato de *output_Gregorian* no satisface sus necesidades, puede utilizar el servicio invocable CEEDATM para convertir *output_seconds* a otro formato.

***fc* (salida)**

Un código de retroalimentación de 12 bytes (opcional) que indica el resultado de este servicio.

Tabla 64. ***Condiciones simbólicas CEELOCT***

Código de comentarios simbólicos	Gravedad	Número de mensaje	texto de mensaje
CEE000	0	--	El servicio se ha completado correctamente.
CEE2F3	3	2531	La hora local no estaba disponible en el sistema.

Notas de uso

- Puede utilizar el servicio invocable CEEGMT para determinar la hora media de Greenwich (GMT).
- Puede utilizar el servicio invocable CEEGMTO para obtener el desplazamiento de GMT a la hora local.
- El valor de carácter devuelto por CEELOCT está diseñado para coincidir con el generado por las funciones intrínsecas existentes. Los valores numéricos devueltos se pueden utilizar para simplificar los cálculos de fecha.

ejemplo

```
*************************************************
**                                             **
** Function: Call CEELOCT to get current       **
**           local time                        **
**                                             **
** In this example, a call is made to CEELOCT  **
** to return the current local time in Lilian  **
** days (the number of days since 14 October   **
** 1582), Lilian seconds (the number of        **
** seconds since 00:00:00 14 October 1582),    **
** and a Gregorian string (in the form         **
** YYYMMDDMISS999). The Gregorian character    **
** string is then displayed.                   **
**                                             **
*************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. CBLLOCT.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  LILIAN                  PIC S9(9) BINARY.
 01  SECONDS                 COMP-2.
 01  GREGORN                 PIC X(17).
 01  FC.
     02  Condition-Token-Value.
     COPY  CEEIGZCT.
         03  Case-1-Condition-ID.
             04  Severity    PIC S9(4) COMP.
             04  Msg-No      PIC S9(4) COMP.
         03  Case-2-Condition-ID
                   REDEFINES Case-1-Condition-ID.
             04  Class-Code  PIC S9(4) COMP.
             04  Cause-Code  PIC S9(4) COMP.
         03  Case-Sev-Ctl    PIC X.
         03  Facility-ID     PIC XXX.
     02  I-S-Info            PIC S9(9) COMP.
 PROCEDURE DIVISION.
 PARA-CBLLOCT.
     CALL 'CEELOCT' USING LILIAN, SECONDS,
```

```
                          GREGORN, FC.
*********************************************
** If CEELOCT runs successfully, display      **
**     Gregorian character string             **
*********************************************
     IF CEE000 of FC  THEN
         DISPLAY 'Local Time is ' GREGORN
     ELSE
         DISPLAY 'CEELOCT failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

     GOBACK.
```

CEEQCEN: consultar la ventana del siglo

CEEQCEN consulta la ventana de siglo, que es un valor de año de dos dígitos.

Cuando desee cambiar la ventana de siglo, utilice CEEQCEN para obtener el valor y, a continuación, utilice CEESCEN para guardar y restaurar el valor actual.

Sintaxis CEEQCEN de CALL

***century_start* (salida)**

Un entero entre 0 y 100 que indica el año en el que se basa la ventana de siglo.

Por ejemplo, si el valor predeterminado de los servicios invocables de Fecha y hora está en vigor, todos los años de dos dígitos se encuentran dentro de la ventana de 100 años que empieza 80 años antes de la fecha del sistema. A continuación, CEEQCEN devuelve el valor 80. Por ejemplo, en el año 2010, 80 indica que todos los años de dos dígitos se encuentran dentro de la ventana de 100 años entre 1930 y 2029, ambos inclusive.

***fc* (salida)**

Un código de retroalimentación de 12 bytes (opcional) que indica el resultado de este servicio.

Tabla 65. ***Condiciones simbólicas de CEEQCEN***

Código de comentarios simbólicos	Gravedad	Número de mensaje	texto de mensaje
CEE000	0	--	El servicio se ha completado correctamente.

ejemplo

```
*********************************************
**                                          **
** Function: Call CEEQCEN to query the      **
**           Fecha y hora callable services **
**           century window                 **
**                                          **
** In this example, CEEQCEN is called to query **
** the date at which the century window starts **
** The century window is the 100-year window   **
** within which the Fecha y hora callable     **
** services assume all two-digit years lie.    **
**                                          **
*********************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. CBLQCEN.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  STARTCW                PIC S9(9) BINARY.
 01  FC.
```

```
     02  Condition-Token-Value.
     COPY  CEEIGZCT.
         03  Case-1-Condition-ID.
             04  Severity    PIC S9(4) COMP.
             04  Msg-No      PIC S9(4) COMP.
         03  Case-2-Condition-ID
                   REDEFINES Case-1-Condition-ID.
             04  Class-Code  PIC S9(4) COMP.
             04  Cause-Code  PIC S9(4) COMP.
         03  Case-Sev-Ctl    PIC X.
         03  Facility-ID     PIC XXX.
     02  I-S-Info            PIC S9(9) COMP.
 PROCEDURE DIVISION.

 PARA-CBLQCEN.
*************************************************
** Call CEEQCEN to return the start of the     **
**     century window                          **
*************************************************

     CALL 'CEEQCEN' USING STARTCW, FC.
*************************************************
** CEEQCEN has no nonzero feedback codes to    **
**     check, so just display result.          **
*************************************************
     IF CEE000 of FC  THEN
         DISPLAY 'The start of the century '
             'window is: ' STARTCW
     ELSE
         DISPLAY 'CEEQCEN failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

     GOBACK.
```

CEESCEN: establecer la ventana del siglo

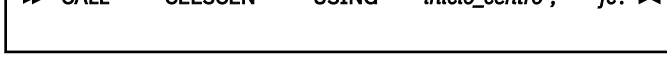
CEESCEN establece la ventana de siglo en un valor de año de dos dígitos para que lo utilicen otros servicios invocables de Fecha y hora .

Utilice CEESCEN junto con CEEDAYS o CEESECS cuando:

- Puede procesar valores de fecha que contienen años de dos dígitos (por ejemplo, en el formato AAMMDD).
- El intervalo de siglo predeterminado no cumple los requisitos de una aplicación determinada.

Para consultar la ventana de siglo, utilice CEEQCEN.

Sintaxis de CALL CEESCEN



inicio_centro
Un entero entre 0 y 100, que establece la ventana de siglo.

Un valor de 80, por ejemplo, coloca todos los años de dos dígitos dentro de la ventana de 100 años que empieza 80 años antes de la fecha del sistema. En 2010, por lo tanto, se supone que todos los años de dos dígitos representan fechas entre 1930 y 2029, inclusive.

***fc* (salida)**
Un código de retroalimentación de 12 bytes (opcional) que indica el resultado de este servicio.

Tabla 66. ***Condiciones simbólicas de CEESCEN***			
Código de comentarios simbólicos	**Gravedad**	**Número de mensaje**	**texto de mensaje**
CEE000	0	--	El servicio se ha completado correctamente.
CEE2E6	3	2502	El UTC/GMT no estaba disponible en el sistema.
CEE2F5	3	2533	El valor pasado a CEESCEN no estaba entre 0 y 100.

ejemplo

```
************************************************
**                                            **
** Function: Call CEESCEN to set the          **
**           Fecha y hora callable services   **
**           century window                   **
**                                            **
** In this example, CEESCEN is called to change **
** the start of the century window to 30 years  **
** before the system date. CEEQCEN is then      **
** called to query that the change made.  A     **
** message that this has been done is then      **
** displayed.                                   **
**                                            **
************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. CBLSCEN.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  STARTCW                PIC S9(9) BINARY.
 01  FC.
     02  Condition-Token-Value.
     COPY  CEEIGZCT.
         03  Case-1-Condition-ID.
             04  Severity    PIC S9(4) COMP.
             04  Msg-No      PIC S9(4) COMP.
         03  Case-2-Condition-ID
                   REDEFINES Case-1-Condition-ID.
             04  Class-Code  PIC S9(4) COMP.
             04  Cause-Code  PIC S9(4) COMP.
         03  Case-Sev-Ctl    PIC X.
         03  Facility-ID     PIC XXX.
     02  I-S-Info            PIC S9(9) COMP.
 PROCEDURE DIVISION.
 PARA-CBLSCEN.
************************************************
** Specify 30 as century start, and two-digit
**     years will be assumed to lie in the
**     100-year window starting 30 years before
**     the system date.
************************************************
     MOVE 30 TO STARTCW.

************************************************
** Call CEESCEN to change the start of the century
**     window.
************************************************
     CALL 'CEESCEN' USING STARTCW, FC.
     IF NOT CEE000 of FC  THEN
         DISPLAY 'CEESCEN failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

 PARA-CBLQCEN.
************************************************
** Call CEEQCEN to return the start of the century
**     window
************************************************
     CALL 'CEEQCEN' USING STARTCW, FC.

************************************************
** CEEQCEN has no nonzero feedback codes to
```

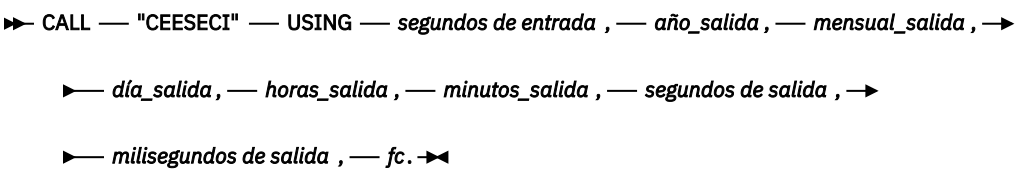
```
**     check, so just display result.
*********************************************
         DISPLAY 'The start of the century '
             'window is: ' STARTCW
     GOBACK.
```

CEESECI: convertir segundos en enteros

CEESECI convierte un número que representa el número de segundos desde 00:00:00 14 de octubre de 1582 en enteros binarios que representan año, mes, día, hora, minuto, segundo y milisegundo.

Utilice CEESECI en lugar de CEEDATM cuando la salida sea necesaria en formato numérico en lugar de en formato de caracteres.

Sintaxis de CALL CEESECI



segundos de entrada
Un número de coma flotante de 64 bits que representa el número de segundos desde las 00:00:00 del 14 de octubre de 1582, sin contar los segundos bisiestos.

Por ejemplo, 00:00:01 el 15 de octubre de 1582 es el segundo número 86,401 (24 * 60 * 60 + 01). El rango de valores válidos para *input_seconds* es de 86,400 a 265.621.679.999,999 (23:59:59.999 31 de diciembre 9999).

Si *input_seconds* no es válido, todos los parámetros de salida excepto el código de retorno se establecen en 0.

***output_year* (salida)**
Un entero binario de 32 bits que representa el año.

El rango de valores válidos para *output_year* es de 1582 a 9999, ambos inclusive.

***output_month* (salida)**
Un entero binario de 32 bits que representa el mes.

El rango de valores válidos para *output_month* es de 1 a 12.

***output_day* (salida)**
Un entero binario de 32 bits que representa el día.

El rango de valores válidos para *output_day* es de 1 a 31.

***output_hours* (salida)**
Un entero binario de 32 bits que representa la hora.

El rango de valores válidos para *output_hours* es de 0 a 23.

***output_minutes* (salida)**
Un entero binario de 32 bits que representa los minutos.

El rango de valores válidos para *output_minutes* es de 0 a 59.

***output_seconds* (salida)**
Un entero binario de 32 bits que representa los segundos.

El rango de valores válidos para *output_seconds* es de 0 a 59.

***output_milisegundos* (salida)**
Entero binario de 32 bits que representa milisegundos.

El rango de valores válidos para *output_milisegundos* es de 0 a 999.

***fc* (salida)**

Un código de retroalimentación de 12 bytes (opcional) que indica el resultado de este servicio.

Tabla 67. ***Condiciones simbólicas de CEESECI***

Código de comentarios simbólicos	Gravedad	Número de mensaje	texto de mensaje
CEE000	0	--	El servicio se ha completado correctamente.
CEE2E9	3	2505	El valor de input_seconds en una llamada a CEEDATM o CEESECI no estaba dentro del rango soportado.

Notas de uso

- El inverso de CEESECI es CEEISEC, que convierte enteros binarios separados que representan año, mes, día, hora, segundo y milisegundo en un número de segundos.
- Si el valor de entrada es una fecha Lilian en lugar de segundos, multiplique la fecha Lilian por 86,400 (número de segundos en un día) y pase el nuevo valor a CEESECI.

ejemplo

```
************************************************
**                                            **
** Function: Call CEESECI to convert seconds   **
**           to integers                       **
**                                            **
** In this example a call is made to CEESECI   **
** to convert a number representing the number **
** of seconds since 00:00:00 14 October 1582   **
** to seven binary integers representing year, **
** month, day, hour, minute, second, and       **
** millisecond.  The results are displayed in  **
** this example.                               **
**                                            **
************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. CBLSECI.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  INSECS                  COMP-2.
 01  YEAR                    PIC S9(9) BINARY.
 01  MONTH                   PIC S9(9) BINARY.
 01  DAYS                    PIC S9(9) BINARY.
 01  HOURS                   PIC S9(9) BINARY.
 01  MINUTES                 PIC S9(9) BINARY.
 01  SECONDS                 PIC S9(9) BINARY.
 01  MILLSEC                 PIC S9(9) BINARY.
 01  IN-DATE.
     02  Vstring-length      PIC S9(4) BINARY.
     02  Vstring-text.
         03  Vstring-char        PIC X,
                     OCCURS 0 TO 256 TIMES
                     DEPENDING ON Vstring-length
                         of IN-DATE.
 01  PICSTR.
     02  Vstring-length      PIC S9(4) BINARY.
     02  Vstring-text.
         03  Vstring-char        PIC X,
                     OCCURS 0 TO 256 TIMES
                     DEPENDING ON Vstring-length
                        of PICSTR.
 01  FC.
     02  Condition-Token-Value.
     COPY  CEEIGZCT.
         03  Case-1-Condition-ID.
             04  Severity    PIC S9(4) COMP.
             04  Msg-No      PIC S9(4) COMP.
         03  Case-2-Condition-ID
                   REDEFINES Case-1-Condition-ID.
```

```
             04  Class-Code  PIC S9(4) COMP.
             04  Cause-Code  PIC S9(4) COMP.
         03  Case-Sev-Ctl    PIC X.
         03  Facility-ID     PIC XXX.
     02  I-S-Info            PIC S9(9) COMP.
 PROCEDURE DIVISION.
 PARA-CBLSECS.
*********************************************
** Call CEESECS to convert time stamp of 6/2/88
**     at 10:23:45 AM to Lilian representation
*********************************************
     MOVE 20 TO Vstring-length of IN-DATE.
     MOVE '06/02/88 10:23:45 AM'
             TO Vstring-text of IN-DATE.
     MOVE 20 TO Vstring-length of PICSTR.
     MOVE 'MM/DD/YY HH:MI:SS AP'
             TO Vstring-text of PICSTR.
     CALL 'CEESECS' USING IN-DATE, PICSTR,
                          INSECS, FC.
     IF NOT CEE000 of FC  THEN
         DISPLAY 'CEESECS failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

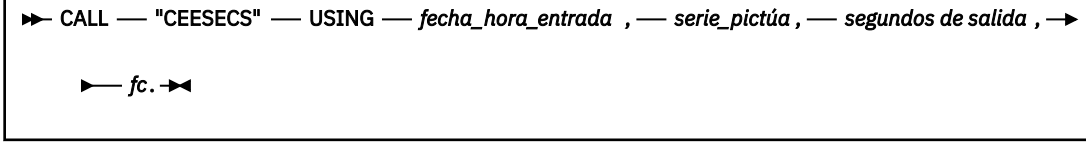
 PARA-CBLSECI.
*********************************************
** Call CEESECI to convert seconds to integers
*********************************************
     CALL 'CEESECI' USING INSECS, YEAR, MONTH,
                          DAYS, HOURS,  MINUTES,
                          SECONDS, MILLSEC, FC.
*********************************************
** If CEESECI runs successfully, display results
*********************************************
     IF CEE000 of FC  THEN
         DISPLAY 'Input seconds of ' INSECS
             ' represents:'
         DISPLAY '   Year......... ' YEAR
         DISPLAY '   Month........ ' MONTH
         DISPLAY '   Day.......... ' DAYS
         DISPLAY '   Hour......... ' HOURS
         DISPLAY '   Minute....... ' MINUTES
         DISPLAY '   Second....... ' SECONDS
         DISPLAY '   Millisecond.. ' MILLSEC
     ELSE
         DISPLAY 'CEESECI failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

     GOBACK.
```

CEESECS: convertir indicación de fecha y hora en segundos

CEESECS convierte una serie que representa una indicación de fecha y hora en Lilian seconds (el número de segundos desde 00:00:00 14 de octubre de 1582). Este servicio facilita la realización de aritmética de tiempo, como por ejemplo el cálculo del tiempo transcurrido entre dos indicaciones de fecha y hora.

Sintaxis de CALL CEESECS



***input_timestamp* (entrada)**

Una serie de caracteres con prefijo de longitud de media palabra que representa una indicación de fecha o hora en un formato que coincide con el especificado por *picture_string*.

La serie de caracteres debe contener entre 5 y 80 caracteres de imagen, ambos incluidos. *input_timestamp* puede contener espacios en blanco iniciales o finales. El análisis empieza con el primer carácter no en blanco (a menos que la propia serie de imagen contenga espacios en

blanco iniciales; en este caso, CEESECS omite exactamente esa cantidad de posiciones antes de que comience el análisis).

Después de analizar una fecha válida, según determine el formato de la fecha que especifique en *picture_string*, CEESECS ignora todos los caracteres restantes. Las fechas válidas están comprendidas entre el 15 de octubre de 1582 y el 31 de diciembre de 9999. Debe especificarse una fecha completa. Las horas válidas van de 00 :00:00.000 a 23 :59:59.999.

Si se omite cualquier parte o todo el valor de tiempo, se sustituyen los ceros por los valores restantes. Por ejemplo:

```
1992-05-17-19:02 is equivalent to 1992-05-17-19:02:00
1992-05-17       is equivalent to 1992-05-17-00:00:00
```

***picture_string* (entrada)**
Una serie de caracteres con prefijo de longitud de media palabra, que indica el formato del valor de fecha o indicación de fecha y hora especificado en *input_timestamp*.

Cada carácter de *picture_string* representa un carácter de *input_timestamp*. Por ejemplo, si especifica MMDDYY HH.MI.SS como *picture_string*, CEESECS lee una *fecha_char_entrada* de 060288 15.35.02 como 3:35:02 PM el 02 de junio de 1988. Si los delimitadores como la barra inclinada (/) aparecen en la serie de imagen, se pueden omitir los ceros iniciales. Por ejemplo, las llamadas siguientes a CEESECS asignan el mismo valor al elemento de datos *segs*:

```
CALL CEESECS USING '92/06/03 15.35.03',
                   'YY/MM/DD HH.MI.SS', secs, fc.
CALL CEESECS USING '92/6/3 15.35.03',
                   'YY/MM/DD HH.MI.SS', secs, fc.
CALL CEESECS USING '92/6/3 3.35.03 PM',
                   'YY/MM/DD HH.MI.SS AP', secs, fc.
CALL CEESECS USING '92.155 3.35.03 pm',
                   'YY.DDD   HH.MI.SS AP', secs, fc.
```

Si *picture_string* incluye un símbolo de era japonesa <JJJJ>, la posición YY en *input_timestamp* representa el número de año dentro de la era japonesa. Por ejemplo, el año 1988 equivale al año japonés 63 en la era Showa.

***output_seconds* (salida)**
Un número de coma flotante de 64 bits que representa el número de segundos desde las 00:00:00 del 14 de octubre de 1582, sin contar los segundos bisiestos. Por ejemplo, 00:00:01 del 15 de octubre de 1582 es el segundo 86,401 (24 * 60 * 60 + 01) en el formato Lilian. 19 :00:01.12 El 16 de mayo de 1988 es el segundo 12,799,191,601.12.

El valor más grande representado es 23 :59:59.999 el 31 de diciembre de 9999, que es el segundo 265.621.679.999,999 en el formato Lilian.

Un valor de coma flotante largo de 64 bits puede representar con precisión aproximadamente 16 dígitos decimales significativos sin pérdida de precisión. Por lo tanto, la precisión está disponible para el milisegundo más cercano (15 dígitos decimales).

Si *input_timestamp* no contiene una indicación de fecha y hora válida, *output_seconds* se establece en 0 y CEESECS termina con un código de retorno simbólico non-CEE000 .

Los cálculos de tiempo transcurrido se realizan fácilmente en *output_seconds*, porque representa el tiempo transcurrido. Las anomalías de año bisiesto y de fin de año no afectan a los cálculos.

***fc* (salida)**
Un código de retroalimentación de 12 bytes (opcional) que indica el resultado de este servicio.

Tabla 68. ***condiciones simbólicas de CEESECS***			
Código de comentarios simbólicos	**Gravedad**	**Número de mensaje**	**texto de mensaje**
CEE000	0	--	El servicio se ha completado correctamente.
CEE2EB	3	2507	No se pasaron datos suficientes a CEEDAYS o CEESECS. El valor Lilian no se ha calculado.
CEE2EC	3	2508	El valor de fecha pasado a CEEDAYS o CEESECS no era válido.
CEE2ED	3	2509	La era pasada a CEEDAYS o CEESECS no fue reconocida.
CEE2EE	3	2510	No se ha reconocido el valor de horas en una llamada a CEEISEC o CEESECS.
CEE2EH	3	2513	La fecha de entrada pasada en una llamada CEEISEC, CEEDAYS o CEESECS no estaba dentro del rango soportado.
CEE2EK	3	2516	No se ha reconocido el valor de minutos en una llamada CEEISEC.
CEE2EL	3	2517	No se ha reconocido el valor de mes en una llamada CEEISEC.
CEE2EM	3	2518	Se ha especificado una serie de imagen no válida en una llamada a un servicio de fecha/hora.
CEE2EN	3	2519	No se ha reconocido el valor de segundos en una llamada CEEISEC.
CEE2EP	3	2521	El valor de año dentro de era < JJJJ>, < CCCC> o < CCCCCCCCCC> pasado a CEEDAYS o CEESECS era cero.
CEE2ET	3	2525	CEESECS ha detectado datos no numéricos en un campo numérico, o la serie de indicación de fecha y hora no coincidía con la serie de imagen.

Notas de uso

- El inverso de CEESECS es CEEDATM, que convierte *output_seconds* al formato de caracteres.
- De forma predeterminada, los años de dos dígitos se encuentran dentro del rango de 100 años que empieza 80 años antes de la fecha del sistema. Así en 2010, todos los años de dos dígitos representan fechas entre 1930 y 2029, inclusive. Puede cambiar este rango utilizando el servicio invocable CEESCEN.

ejemplo

```
*************************************************
**                                             **
** Function: Call CEESECS to convert           **
**           time stamp to number of seconds   **
**                                             **
** In this example, calls are made to CEESECS  **
** to convert two time stamps to the number    **
** of seconds since 00:00:00 14 October 1582.  **
** The Lilian seconds for the earlier          **
** time stamp are then subtracted from the     **
** Lilian seconds for the later time stamp     **
** to determine the number of between the      **
** two.  This result is displayed.             **
**                                             **
*************************************************
 IDENTIFICATION DIVISION.
```

```
 PROGRAM-ID. CBLSECS.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  SECOND1                 COMP-2.
 01  SECOND2                 COMP-2.
 01  TIMESTP.
     02  Vstring-length      PIC S9(4) BINARY.
     02  Vstring-text.
         03  Vstring-char        PIC X,
                     OCCURS 0 TO 256 TIMES
                     DEPENDING ON Vstring-length
                        of TIMESTP.
 01  TIMESTP2.
     02  Vstring-length      PIC S9(4) BINARY.
     02  Vstring-text.
         03  Vstring-char        PIC X,
                     OCCURS 0 TO 256 TIMES
                     DEPENDING ON Vstring-length
                        of TIMESTP2.
 01  PICSTR.
     02  Vstring-length      PIC S9(4) BINARY.
     02  Vstring-text.
         03  Vstring-char        PIC X,
                     OCCURS 0 TO 256 TIMES
                     DEPENDING ON Vstring-length
                        of PICSTR.
 01  FC.
     02  Condition-Token-Value.
     COPY  CEEIGZCT.
         03  Case-1-Condition-ID.
             04  Severity    PIC S9(4) COMP.
             04  Msg-No      PIC S9(4) COMP.
         03  Case-2-Condition-ID
                   REDEFINES Case-1-Condition-ID.
             04  Class-Code  PIC S9(4) COMP.
             04  Cause-Code  PIC S9(4) COMP.
         03  Case-Sev-Ctl    PIC X.
         03  Facility-ID     PIC XXX.
     02  I-S-Info            PIC S9(9) COMP.
 PROCEDURE DIVISION.

 PARA-SECS1.
************************************************
** Specify first time stamp and a picture string
**     describing the format of the time stamp
**     as input to CEESECS
************************************************
     MOVE 25 TO Vstring-length of TIMESTP.
     MOVE '1969-05-07 12:01:00.000'
             TO Vstring-text of TIMESTP.
     MOVE 25 TO Vstring-length of PICSTR.
     MOVE 'YYYY-MM-DD HH:MI:SS.999'
             TO Vstring-text of PICSTR.

************************************************
** Call CEESECS to convert the first time stamp
** to Lilian seconds
************************************************
     CALL 'CEESECS' USING TIMESTP, PICSTR,
                          SECOND1, FC.
     IF NOT CEE000 of FC  THEN
         DISPLAY 'CEESECS failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

 PARA-SECS2.
************************************************
** Specify second time stamp and a picture string
**     describing the format of the time stamp as
**     input to CEESECS.
************************************************
     MOVE 25 TO Vstring-length of TIMESTP2.
     MOVE '2004-01-01 00:00:01.000'
             TO Vstring-text of TIMESTP2.
     MOVE 25 TO Vstring-length of PICSTR.
     MOVE 'YYYY-MM-DD HH:MI:SS.999'
             TO Vstring-text of PICSTR.

************************************************
** Call CEESECS to convert the second time stamp
```

```
**     to Lilian seconds
*********************************************
     CALL 'CEESECS' USING TIMESTP2, PICSTR,
                          SECOND2, FC.
     IF NOT CEE000 of FC  THEN
         DISPLAY 'CEESECS failed with msg '
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.

 PARA-SECS2.
*********************************************
** Subtract SECOND2 from SECOND1 to determine the
**     number of seconds between the two time stamps
*********************************************
     SUBTRACT SECOND1 FROM SECOND2.
     DISPLAY 'The number of seconds between '
         Vstring-text OF TIMESTP ' and '
         Vstring-text OF TIMESTP2 ' is: ' SECOND2.

     GOBACK.
```

"Ejemplo: series de imágenes de fecha y hora" en la página 545

Referencias relacionadas
"Términos y series de caracteres de imagen" en la página 542

CEEUTC: obtener hora universal coordinada

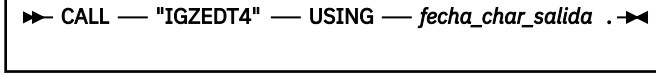
CEEUTC es idéntico a CEEGMT.

Referencias relacionadas
"CEEGMT: obtener hora media de Greenwich actual" en la página 578

IGZEDT4: obtener fecha actual

IGZEDT4 devuelve la fecha actual con un año de cuatro dígitos con el formato AAAAMMDD.

Sintaxis de CALL IGZEDT4



***output_char_date* (salida)**
Serie de caracteres de longitud fija de 8 bytes con el formato AAAAMMDD, que representa el año, mes y día actuales.

Nota de uso: IGZEDT4 no está soportado en CICS.

ejemplo

```
*************************************************
** Function: IGZEDT4 - get current date in the  **
**                     format YYYYMMDD.         **
*************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. CBLEDT4.
 . . .
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  CHRDATE                PIC S9(8) USAGE DISPLAY.
 . . .
 PROCEDURE DIVISION.
 PARA-CBLEDT4.
*************************************************
** Call IGZEDT4.
*************************************************
     CALL 'IGZEDT4' USING BY REFERENCE CHRDATE.
*************************************************
** IGZEDT4 has no nonzero return code to
**     check, so just display result.
```

```
************************************************
     DISPLAY 'The current date is: '
         CHRDATE
     GOBACK.
```

Como alternativa, puede utilizar la sentencia ACCEPT para obtener la fecha actual con un año de cuatro dígitos. A continuación se muestra un ejemplo:

```
01 todays-date.
03 todays-yyyy pic 9(04).
03 todays-mm pic 9(02).
03 todays-dd pic 9(02).
. . .
accept todays-date from date yyyymmdd.
```

Apéndice D. Material de referencia XML

La información siguiente describe los códigos de excepción XML que se pueden devolver durante el análisis XML o la generación XML. La información también documenta las restricciones de bienestar del *Especificación XML* que comprueba el analizador.

Referencias relacionadas
“Excepciones XML PARSE” en la página 597
“Conformidad XML” en la página 606
“excepciones XML GENERATE” en la página 608
Especificación XML

Excepciones XML PARSE

Cuando se produce un suceso de excepción, el analizador XML establece el registro especial XML-CODE en un valor que identifica la excepción. En función del valor de XML-CODE, es posible que el analizador pueda continuar o no el proceso después de la excepción, tal como se detalla en la información a la que se hace referencia a continuación.

Referencias relacionadas
“Excepciones XML PARSE que permiten la continuación” en la página 597
“Excepciones XML PARSE que no permiten la continuación” en la página 602

Excepciones XML PARSE que permiten la continuación

Si el analizador XML puede continuar el proceso después de un suceso de excepción depende del valor del código de excepción.

El analizador puede continuar procesando si el código de excepción, que está en el registro especial XML-CODE, está dentro de uno de los rangos siguientes:

- 1 - 99
- 100,001 - 165,535
- 200,001 - 265,535

La tabla siguiente describe cada excepción e identifica las acciones que el analizador realiza si solicita que continúe después de la excepción. Algunas de las descripciones utilizan los términos siguientes:

- *Codificación de documento real*
- *Declaración de codificación de documentos*
- *Página de códigos ASCII externo*
- *Página de códigos EBCDIC externa*

Para ver las definiciones de los términos, consulte el concepto relacionado sobre la codificación de documentos de entrada XML.

Tabla 69. ***Excepciones XML PARSE que permiten la continuación***		
Código de excepción (decimal)	**Descripción**	**Acción de analizador en continuación**
1	El analizador ha encontrado un carácter no válido al explorar el espacio en blanco fuera del contenido del elemento. Para obtener más información sobre el espacio en blanco, consulte el concepto relacionado sobre la codificación de documentos de entrada XML.	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.
2	El analizador ha encontrado un inicio no válido de una instrucción de proceso, elemento, comentario o declaración de tipo de documento fuera del contenido del elemento.	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.
3	El analizador ha encontrado un nombre de atributo duplicado.	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.
4	El analizador ha encontrado el carácter de marcación '<' en un valor de atributo.	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.
5	Los nombres de etiqueta de inicio y final de un elemento no coinciden.	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.
6	El analizador ha encontrado un carácter no válido en el contenido del elemento.	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.
7	El analizador ha encontrado un inicio no válido de un elemento, comentario, instrucción de proceso o sección CDATA en el contenido del elemento.	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.
8	El analizador ha encontrado en el contenido del elemento la secuencia de caracteres de cierre de CDATA ']] >' sin la secuencia de caracteres de apertura coincidente '<![CDATA ['.	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.

Tabla 69. ***Excepciones XML PARSE que permiten la continuación*** *(continuación)*

Código de excepción (decimal)	**Descripción**	**Acción de analizador en continuación**
9	El analizador ha encontrado un carácter no válido en un comentario.	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.
10	El analizador ha encontrado en un comentario la secuencia de caracteres '- -' (dos guiones) no seguida de '>'.	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.
11	El analizador ha encontrado un carácter no válido en un segmento de datos de instrucción de proceso.	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.
12	La declaración XML no estaba al principio del documento.	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.
13	El analizador ha encontrado un dígito no válido en una referencia de carácter hexadecimal (con el formato &#x*dddd*;).	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.
14	El analizador ha encontrado un dígito no válido en una referencia de carácter decimal (con el formato &#*dddd*;).	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.
15	El valor de la declaración de codificación de la declaración XML no empezaba por las minúsculas ni por las mayúsculas de la A a la Z.	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.
16	Una referencia de carácter no hacía referencia a un carácter XML permitido.	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.
17	El analizador ha encontrado un carácter no válido en un nombre de referencia de entidad.	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT`.

*Tabla 69. **Excepciones XML PARSE que permiten la continuación** (continuación)*		
Código de excepción (decimal)	**Descripción**	**Acción de analizador en continuación**
18	El analizador ha encontrado un carácter no válido en un valor de atributo.	El analizador continúa detectando errores hasta que alcanza el final del documento o encuentra un error que no permite la continuación. El analizador no señala ningún suceso normal adicional, excepto el suceso `END-OF-DOCUMENT` .
70	La codificación de documento real era EBCDIC, y la opción de compilador página de códigos EBCDIC externa está soportada, pero la declaración de codificación de documento no especificaba una página de códigos EBCDIC soportada.	El analizador utiliza la codificación especificada por la opción de compiladorpágina de códigos EBCDIC externa.
71	La codificación de documento real era EBCDIC, y la declaración de codificación de documento especificaba una codificación EBCDIC soportada, pero la opción de compilador la página de códigos EBCDIC externa no está soportada.	El analizador utiliza la codificación especificada por la declaración de codificación de documento.
72	La codificación de documento real era EBCDIC, la opción de compilador la página de códigos EBCDIC externa no está soportaday el documento no contenía una declaración de codificación.	El analizador utiliza la página de códigos EBCDIC 1140 (EE.UU., Canadá,... Página de códigos ampliada del país del euro).
73	La codificación de documento real era EBCDIC, pero ni la opción de compilador página de códigos EBCDIC externa ni la declaración de codificación de documento especificaban una página de códigos EBCDIC soportada.	El analizador utiliza la página de códigos EBCDIC 1140 (EE.UU., Canadá,... Página de códigos ampliada del país del euro).
80	La codificación del documento real era ASCII, y la página de códigos ASCII externa está soportada, pero la declaración de codificación del documento no especificaba una página de códigos ASCII soportada.	El analizador utiliza la codificación especificada por la página de códigos ASCII externa.
81	La codificación de documento real era ASCII, y la declaración de codificación de documento especificaba una codificación ASCII soportada, pero la página de códigos ASCII externa no está soportada.	El analizador utiliza la codificación especificada por la declaración de codificación de documento.

Tabla 69. ***Excepciones XML PARSE que permiten la continuación*** *(continuación)*

Código de excepción (decimal)	**Descripción**	**Acción de analizador en continuación**
82	La codificación del documento real era ASCII, pero la página de códigos ASCII externa no está soportada y el documento no contenía una declaración de codificación.	El analizador utiliza la página de códigos ASCII 819 (ISO-8859-1 Latin 1/Open Systems).
83	La codificación del documento real era ASCII, pero ni la página de códigos ASCII externa ni la declaración de codificación del documento especificaban una página de códigos ASCII soportada.	El analizador utiliza la página de códigos ASCII 819 (ISO-8859-1 Latin 1/Open Systems).
84	La codificación de documento real era ASCII pero no era válida UTF-8, la página de códigos externa especificaba UTF-8y el documento no contenía una declaración de codificación.	El analizador utiliza UTF-8.
85	La codificación de documento real era ASCII pero no válida UTF-8, la página de códigos externa especificaba UTF-8y la declaración de codificación de documento no especificaba ni una página de códigos ASCII soportada ni UTF-8.	El analizador utiliza UTF-8.
86	La codificación de documento real era ASCII pero no válida UTF-8, la página de códigos externa especificaba una página de códigos ASCII soportada y la declaración de codificación de documento especificaba UTF-8.	El analizador utiliza UTF-8.
87	La codificación de documento real era ASCII pero no era válida UTF-8, y la página de códigos externa y la declaración de codificación de documento especificaban UTF-8.	El analizador utiliza UTF-8.
88	La codificación de documento real era ASCII pero no válida UTF-8, la página de códigos externa no especificaba ni una página de códigos ASCII soportada ni UTF-8y la declaración de codificación de documento especificaba UTF-8.	El analizador utiliza UTF-8.

*Tabla 69. **Excepciones XML PARSE que permiten la continuación** (continuación)*		
Código de excepción (decimal)	**Descripción**	**Acción de analizador en continuación**
89	La codificación de documento real era ASCII pero no válida UTF-8, la página de códigos externa especificaba UTF-8y la declaración de codificación de documento especificaba una página de códigos ASCII soportada.	El analizador utiliza UTF-8.
92	El elemento de datos de documento era alfanumérico, pero la codificación de documento real era Unicode UTF-16.	El analizador utiliza la página de códigos 1200 (Unicode UTF-16).
100,001 - 165,535	La página de códigos EBCDIC externa y la declaración de codificación de documento han especificado distintas páginas de códigos EBCDIC soportadas. XML-CODE contiene el CCSID de la página de códigos para la declaración de codificación más 100.000.	Si establece XML-CODE en cero antes de volver del suceso EXCEPTION , el analizador utiliza la codificación especificada por la página de códigos EBCDIC externa. Si establece XML-CODE en el CCSID para la declaración de codificación de documento (restando 100.000), el analizador utiliza esta codificación.
200,001 - 265,535	La página de códigos ASCII externa y la declaración de codificación de documento han especificado distintas páginas de códigos ASCII soportadas. XML-CODE contiene el CCSID para la declaración de codificación más 200.000.	Si establece XML-CODE en cero antes de volver del suceso EXCEPTION , el analizador utiliza la codificación especificada por la página de códigos ASCII externa. Si establece XML-CODE en el CCSID para la declaración de codificación de documento (restando 200.000), el analizador utiliza esta codificación.

Conceptos relacionados
"CÓDIGO XML" en la página 424
"Codificación de documento de entrada XML" en la página 427

Tareas relacionadas
"Manejo de excepciones XML PARSE" en la página 430

Excepciones XML PARSE que no permiten la continuación

El analizador XML no puede continuar el proceso si se produce alguna de las excepciones descritas a continuación.

No se devuelven más sucesos del analizador para ninguna de estas excepciones incluso si el procedimiento de proceso establece XML-CODE en cero antes de volver a pasar el control al analizador. El analizador transfiere el control a la sentencia en la frase ON EXCEPTION , si se especifica, de lo contrario, al final de la sentencia XML PARSE .

*Tabla 70. **Excepciones XML PARSE que no permiten la continuación***	
Código de excepción (decimal)	**Descripción**
100	El analizador ha alcanzado el final del documento al explorar el inicio de la declaración XML.

Tabla 70. ***Excepciones XML PARSE que no permiten la continuación*** *(continuación)*	
Código de excepción (decimal)	**Descripción**
101	El analizador ha alcanzado el final del documento mientras buscaba el final de la declaración XML.
102	El analizador ha llegado al final del documento mientras buscaba el elemento raíz.
103	El analizador ha alcanzado el final del documento mientras buscaba la información de versión en la declaración XML.
104	El analizador ha llegado al final del documento mientras buscaba el valor de información de versión en la declaración XML.
106	El analizador ha llegado al final del documento mientras buscaba el valor de declaración de codificación en la declaración XML.
108	El analizador ha alcanzado el final del documento mientras buscaba el valor de declaración `standalone` en la declaración XML.
109	El analizador ha alcanzado el final del documento al explorar un nombre de atributo.
110	El analizador ha alcanzado el final del documento al explorar un valor de atributo.
111	El analizador ha alcanzado el final del documento al explorar una referencia de carácter o una referencia de entidad en un valor de atributo.
112	El analizador ha alcanzado el final del documento al explorar un código de elemento vacío.
113	El analizador ha alcanzado el final del documento al explorar el nombre del elemento raíz.
114	El analizador ha alcanzado el final del documento al explorar un nombre de elemento.
115	El analizador ha alcanzado el final del documento al explorar los datos de caracteres en el contenido del elemento.
116	El analizador ha llegado al final del documento al explorar una instrucción de proceso en el contenido del elemento.
117	El analizador ha llegado al final del documento al explorar un comentario o sección CDATA en el contenido del elemento.
118	El analizador ha alcanzado el final del documento al explorar un comentario en el contenido del elemento.
119	El analizador ha llegado al final del documento al explorar una sección CDATA en el contenido del elemento.
120	El analizador ha alcanzado el final del documento al explorar una referencia de carácter o una referencia de entidad en el contenido del elemento.
121	El analizador ha alcanzado el final del documento al explorar después del cierre del elemento raíz.
122	El analizador ha encontrado un posible inicio no válido de una declaración de tipo de documento.
123	El analizador ha encontrado una segunda declaración de tipo de documento.
124	El primer carácter del nombre del elemento raíz no era una letra, '_' o ':'.

*Tabla 70. **Excepciones XML PARSE que no permiten la continuación** (continuación)*	
Código de excepción (decimal)	**Descripción**
125	El primer carácter del primer nombre de atributo de un elemento no era una letra, '_' o ':'.
126	El analizador ha encontrado un carácter no válido en o después de un nombre de elemento.
127	El analizador ha encontrado un carácter distinto de '=' a continuación de un nombre de atributo.
128	El analizador ha encontrado un delimitador de valor de atributo no válido.
130	El primer carácter de un nombre de atributo no era una letra, '_' o ':'.
131	El analizador ha encontrado un carácter no válido en o después de un nombre de atributo.
132	Un código de elemento vacío no ha terminado en '>' después de '/'.
133	El primer carácter de un nombre de etiqueta de final de elemento no era una letra, '_' o ':'.
134	Un nombre de etiqueta final de elemento no ha terminado con un '>'.
135	El primer carácter de un nombre de elemento no era una letra, '_' o ':'.
136	El analizador ha encontrado un inicio no válido de un comentario o sección CDATA en el contenido del elemento.
137	El analizador ha encontrado un inicio no válido de un comentario.
138	El primer carácter de un nombre de destino de instrucción de proceso no era una letra, '_' o ':'.
139	El analizador ha encontrado un carácter no válido en o después de un nombre de destino de instrucción de proceso.
140	La secuencia de caracteres de cierre '?>' no ha terminado una instrucción de proceso.
141	El analizador ha encontrado un carácter no válido después de '&' en una referencia de carácter o referencia de entidad.
142	La información de versión no estaba presente en la declaración XML.
143	'version' en la declaración XML no estaba seguido por '='.
144	El valor de declaración de versión en la declaración XML falta o está delimitado incorrectamente.
145	El valor de información de versión de la declaración XML ha especificado un carácter incorrecto o los delimitadores de inicio y final no coinciden.
146	El analizador ha encontrado un carácter no válido después del delimitador de cierre de valor de información de versión en la declaración XML.
147	El analizador ha encontrado un atributo no válido en lugar de la declaración de codificación opcional en la declaración XML.
148	'encoding' en la declaración XML no estaba seguido por '='.
149	El valor de la declaración de codificación de la declaración XML falta o está delimitado incorrectamente.

Tabla 70. ***Excepciones XML PARSE que no permiten la continuación*** *(continuación)*	
Código de excepción (decimal)	**Descripción**
150	El valor de declaración de codificación de la declaración XML ha especificado un carácter incorrecto o los delimitadores de inicio y final no coinciden.
151	El analizador ha encontrado un carácter no válido después del delimitador de cierre de valor de declaración de codificación en la declaración XML.
152	El analizador ha encontrado un atributo no válido en lugar de la declaración `standalone` opcional en la declaración XML.
153	`standalone` en la declaración XML no estaba seguido por =.
154	El valor de la declaración `standalone` en la declaración XML falta o está delimitado incorrectamente.
155	El valor de la declaración `standalone` no era solo 'yes' ni 'no'.
156	El valor de declaración `standalone` de la declaración XML ha especificado un carácter incorrecto o los delimitadores de inicio y final no coinciden.
157	El analizador ha encontrado un carácter no válido después del delimitador de cierre del valor de declaración `standalone` en la declaración XML.
158	La declaración XML no ha terminado por la secuencia de caracteres adecuada '?>', o contenía un atributo no válido.
159	El analizador ha encontrado el inicio de una declaración de tipo de documento después del final del elemento raíz.
160	El analizador ha encontrado el inicio de un elemento después del final del elemento raíz.
161	El analizador ha encontrado una secuencia de bytes UTF-8 no válida.
162	El analizador ha encontrado un carácter UTF-8 que tiene un valor escalar Unicode mayor que x 'FFFF'.
315	La *codificación de documento real* era UTF-16 little-endian, a la que el analizador no da soporte en esta plataforma.
316	La codificación de documento real era UCS4, a la que el analizador no da soporte.
317	El analizador no puede determinar la codificación del documento. Es posible que el documento esté dañado.
318	La codificación de documento real era UTF-8, a la que el analizador no da soporte.
320	El elemento de datos del documento era nacional, pero la codificación del documento real era EBCDIC.
321	El elemento de datos de documento era nacional, pero la codificación de documento real era ASCII.
322	El elemento de datos del documento era un elemento de datos alfanumérico nativo, pero la codificación del documento real era EBCDIC.
323	El elemento de datos del documento era un elemento de datos alfanumérico del sistema principal, pero la codificación del documento real era ASCII.
324	El elemento de datos del documento era nacional, pero la codificación del documento real era UTF-8.

Tabla 70. ***Excepciones XML PARSE que no permiten la continuación*** *(continuación)*	
Código de excepción (decimal)	**Descripción**
325	El elemento de datos de documento era un elemento de datos alfanumérico de host, pero la codificación de documento real era UTF-8.
500 - 599	Error interno. Informe del error al representante de servicio.

Conceptos relacionados
"CÓDIGO XML" en la página 424

Tareas relacionadas
"Manejo de excepciones XML PARSE" en la página 430

Conformidad XML

El analizador COBOL XML incorporado que se incluye en COBOL para Linux no es un procesador XML compatible según la definición de *Especificación XML*. El analizador no valida los documentos XML que analiza. Aunque comprueba si hay muchos errores de buena calidad, no realiza todas las acciones necesarias para un procesador XML no validador.

En concreto, el analizador no procesa la definición de tipo de documento interno (subconjunto interno DTD). Por lo tanto, no proporciona valores de atributo predeterminados, no normaliza los valores de atributo y no incluye el texto de sustitución de entidades internas excepto para las entidades predefinidas. En su lugar, pasa toda la declaración de tipo de documento como el contenido de `XML-TEXT` o `XML-NTEXT` para el suceso XML `DOCUMENT-TYPE-DECLARATION` , lo que permite a la aplicación realizar estas acciones si es necesario.

El analizador permite opcionalmente que los programas continúen procesando un documento XML después de algunos errores. La finalidad de permitir que el proceso continúe es facilitar la depuración de documentos XML y procedimientos de proceso.

Recapitulando la definición en *Especificación XML*, un objeto textual es un documento XML *bien formado* si:

- Tomado como un todo, se ajusta a la gramática para documentos XML.
- Cumple todas las restricciones de bienestar explícitas listadas en la publicación *Especificación XML*.
- Cada entidad analizada (fragmento de texto) a la que se hace referencia directa o indirectamente dentro del documento está bien formada.

El analizador COBOL XML comprueba que los documentos se ajustan a la gramática XML, excepto para cualquier declaración de tipo de documento. La declaración se proporciona en su totalidad, sin comprobar, a su solicitud.

La información siguiente es una anotación de *Especificación XML*. El W3C no es responsable de ningún contenido que no se encuentre en el URL original (www.w3.org/TR/REC-xml). Todas las anotaciones son no normativas y se muestran en *cursiva*.

La *Especificación XML* también contiene doce restricciones de bienestar explícitas. Las restricciones que el analizador XML COBOL comprueba parcial o completamente se muestran en el tipo **negrita** :

1. Entidades de parámetro (PE) en subconjunto interno: "En el subconjunto DTD interno, las referencias de entidad de parámetro sólo pueden producirse cuando las declaraciones de marcación pueden

producirse, no dentro de las declaraciones de marcación. (Esto no se aplica a las referencias que se producen en entidades de parámetros externos o al subconjunto externo.)"

El analizador no procesa el subconjunto DTD interno, por lo que no aplica esta restricción.

2. Subconjunto externo: "El subconjunto externo, si lo hay, debe coincidir con la producción de extSubset."

   El analizador no procesa el subconjunto externo, por lo que no aplica esta restricción.

3. Entidad de parámetro entre declaraciones: "El texto de sustitución de una referencia de entidad de parámetro en un DeclSep debe coincidir con el extSubsetDecl de producción."

   El analizador no procesa el subconjunto DTD interno, por lo que no aplica esta restricción.

4. **Coincidencia de tipo de elemento:** "El nombre de la etiqueta final de un elemento debe coincidir con el tipo de elemento de la etiqueta inicial."

   El analizador aplica esta restricción.

5. **Especificación de atributo exclusivo:** "Ningún nombre de atributo puede aparecer más de una vez en el mismo código de inicio o código de elemento vacío."

   El analizador soporta parcialmente esta restricción comprobando la exclusividad de hasta 10 nombres de atributo en un elemento determinado. La aplicación puede comprobar cualquier nombre de atributo que supere este límite.

6. Sin referencias de entidad externa: "Los valores de atributo no pueden contener referencias de entidad directas o indirectas a entidades externas."

   El analizador no aplica esta restricción.

7. No '<' en Valores de atributo: "El texto de sustitución de cualquier entidad a la que se haga referencia directa o indirectamente en un valor de atributo no debe contener un '<'."

   El analizador no aplica esta restricción.

8. **Carácter legal:** "Los caracteres a los que se hace referencia utilizando referencias de caracteres deben coincidir con la producción de Char."

   El analizador aplica esta restricción.

9. Entidad declarada: "En un documento sin ninguna DTD, un documento con sólo un subconjunto DTD interno que no contiene referencias de entidad de parámetro, o un documento con standalone = 'yes', para una referencia de entidad que no se produce dentro del subconjunto externo o una entidad de parámetro, el nombre proporcionado en la referencia de entidad debe coincidir con el de una declaración de entidad que no se produce dentro del subconjunto externo o una entidad de parámetro, salvo que los documentos bien formados no tengan que declarar ninguna de las siguientes entidades: amp, lt, gt, apos, quot. La declaración de una entidad general debe preceder a cualquier referencia a la misma que aparezca en un valor predeterminado en una declaración de lista de atributos."

   "Tenga en cuenta que si las entidades se declaran en el subconjunto externo o en las entidades de parámetros externos, un procesador no validador no está obligado a leer y procesar sus declaraciones; para dichos documentos, la regla de que una entidad debe declararse es una restricción de integridad sólo si es autónoma = 'yes'."

   El analizador no aplica esta restricción.

10. Entidad analizada: "Una referencia de entidad no debe contener el nombre de una entidad no analizada. Sólo se puede hacer referencia a entidades no analizadas en valores de atributo declarados de tipo ENTITY o ENTITIEs."

    El analizador no aplica esta restricción.

11. Sin recurrencia: "Una entidad analizada no debe contener una referencia recursiva a sí misma, ya sea directa o indirectamente."

    El analizador no aplica esta restricción.

12. En DTD: "Las referencias de entidad de parámetros sólo pueden aparecer en la DTD."

El analizador no impone esta restricción, porque no se puede producir el error.

El material anterior es una anotación del *Especificación XML*. El W3C no es responsable de ningún contenido que no se encuentre en el URL original (www.w3.org/TR/REC-xml); todas estas anotaciones no son normativas. Este documento ha sido revisado por W3C Miembros y otras partes interesadas y ha sido aprobado por el Director como Recomendación W3C . Es un documento estable y puede ser utilizado como material de referencia o citado como referencia normativa de otro documento. La versión normativa de la especificación es la versión en inglés que se encuentra en el sitio W3C ; cualquier documento traducido puede contener errores de la traducción.

Conceptos relacionados
"Analizador XML en COBOL" en la página 419

Referencias relacionadas
XML (Extensible Markup Language)
Especificación XML (Prolog y declaración de tipo de documento)

excepciones XML GENERATE

Es posible que se devuelva uno de varios códigos de excepción en el registro especial `XML-CODE` durante la generación de XML. Si se produce una de estas excepciones, el control se pasa a la sentencia en la frase `ON EXCEPTION` , o al final de la sentencia `XML GENERATE` si no ha codificado una frase `ON EXCEPTION` .

Tabla 71. ***Excepciones de XML GENERATE***

Código de excepción (decimal)	**Descripción**
400	El receptor era demasiado pequeño para contener el documento XML generado. El elemento de datos `COUNT IN` , si se especifica, contiene el recuento de posiciones de caracteres que se han generado realmente.
401	Un nombre de datos multibyte contenía un carácter que, cuando se convertía a Unicode, no era válido en un nombre de elemento o atributo XML.
402	El primer carácter de un nombre de datos multibyte , cuando se convirtió a Unicode, no era válido como primer carácter de un nombre de elemento o atributo de XML.
403	El valor de una variable `OCCURS DEPENDING ON` ha superado los 16.777.215.
410	La página página CCSID especificada por la variable de entorno EBCDIC_CODEPAGE no está soportada para la conversión a Unicode.
411	El CCSID CCSID especificado por la variable de entorno EBCDIC_CODEPAGE no es un página de códigosEBCDIC de un solo byte soportado.
412	El receptor era alfanumérico nativo, pero la codificación especificada para el documento no era UTF-8 o una página de códigos ASCII de un solo byte soportada.
413	El receptor era alfanumérico, pero el entorno local de tiempo de ejecución no era coherente con el entorno local de tiempo de compilación.
414	La especificada para el documento XML no era válida o no estaba una página de códigos soportada.
415	El receptor era nacional, pero la codificación especificada para el documento no era UTF-16.
416	El identificador de espacio de nombres XML contenía caracteres XML no válidos.

*Tabla 71. **Excepciones de XML GENERATE** (continuación)*	
Código de excepción (decimal)	**Descripción**
417	El contenido de carácter de elemento o un valor de atributo contenía caracteres que no están permitidos en el contenido XML. La generación XML ha continuado, con el nombre de código de elemento o el nombre de atributo con el prefijo 'hex.' y el valor de datos original representado en el documento en hexadecimal.
418	La conversión de codificación ha generado caracteres de sustitución.
419	El prefijo de espacio de nombres XML no era válido.
420	El elemento de datos de origen incluía un nombre de varios bytes o contenido de varios bytes, y el receptor era alfanumérico nativo, pero la codificación especificada para el documento no era UTF-8.
600-699	Error interno. Informe del error al representante de servicio.

Tareas relacionadas

“Manejo de excepciones XML GENERATE” en la página 445

Referencias relacionadas

Apéndice E. Opción de compilador EXIT

Puede utilizar la opción de compilador EXIT para proporcionar módulos proporcionados por el usuario en lugar de varias funciones de compilador. Para obtener detalles sobre el proceso de cada módulo de salida, el manejo de errores para módulos de salida o el uso de la opción EXIT con sentencias CICS y SQL , consulte los temas siguientes.

Referencias relacionadas
"Área de trabajo de salida de usuario y extensión de área de trabajo" en la página 611
"Lista de parámetros para módulos de salida" en la página 611
"Procesamiento de INEXIT" en la página 614
"Procesamiento de LIBEXIT" en la página 615
"Proceso de PRTEXIT" en la página 615

"Proceso de MSGEXIT" en la página 616
"Manejo de errores para módulos de salida" en la página 623
"salida" en la página 282

Área de trabajo de salida de usuario y extensión de área de trabajo

Cuando se utiliza una de las salidas de usuario, el compilador proporciona un área de trabajo y una extensión de área de trabajo en la que puede guardar la dirección de almacenamiento obtenida por el módulo de salida. Tener tales áreas de trabajo permite que el módulo sea reentrante.

El área de trabajo de salida de usuario (para que lo utilicen INEXIT, LIBEXIT y PRTEXIT) consta de 4 palabras completas que residen en un límite de palabra completa. La extensión de área de trabajo de salida de usuario (para que la utilice MSGEXIT) consta de 8 palabras completas también en un límite de palabra completa. Estas palabras completas se inicializan en ceros binarios antes de que se invoque la primera rutina de salida y se pasan al módulo de salida en una lista de parámetros. Después de la inicialización, el compilador no hace más referencia a las áreas de trabajo.

Referencias relacionadas
"Procesamiento de INEXIT" en la página 614
"Procesamiento de LIBEXIT" en la página 615
"Proceso de PRTEXIT" en la página 615

"Proceso de MSGEXIT" en la página 616

Lista de parámetros para módulos de salida

El compilador utiliza una estructura, pasada por referencia, para comunicarse con el módulo de salida.

Tabla 72. ***Lista de parámetros para módulos de salida***

Parámetro	**Descripción del elemento**
Tipo de salida de usuario	Media palabra que identifica qué salida de usuario debe realizar la operación: • 1=INEXIT • 2=LIBEXIT • 3=PRTEXIT • 5=Reserved • 6=MSGEXIT

Tabla 72. ***Lista de parámetros para módulos de salida*** *(continuación)*	
Parámetro	**Descripción del elemento**
Código de operación	Media palabra que indica el tipo de operación: • 0=OPEN • 1=CLOSE • 2=GET • 3=PUT • 4=FIND • 5=MSGSEV: personalizar gravedad de mensaje
Código de retorno	Palabra completa, establecida por el módulo de salida, que indica el éxito de la operación solicitada. Para los códigos de operación 0 a 4: • 0=Successful • 4=End-of-data • 12=Failed Para el código de operación 5: • 0=Message no personalizado • 4=Message encontrado y personalizado • 12=Operation ha fallado
Longitud de registro	Palabra completa, establecida por el módulo de salida, que indica la longitud del registro devuelto por la operación GET, o proporcionada por la operación PUT.
Dirección del registro o *str2*	En modalidad de 32 bits: Palabra completa, establecida por el módulo de salida en la dirección del registro en un almacenamiento intermedio propiedad del usuario para la operación GET, o establecida por el compilador en la dirección del registro de la operación PUT. *str2* sólo se aplica a OPEN. La primera media palabra (en un límite de media palabra) contiene la longitud de la serie, seguida de la serie. En modalidad de 64 bits: Palabra doble (8 bytes).
Área de trabajo de salida de usuario	Área de trabajo de cuatro palabras completas proporcionada por el compilador para que lo utilice el módulo de salida de usuario: • Primera palabra: para uso de INEXIT • Segunda palabra: para uso de LIBEXIT • Tercera palabra: para uso de PRTEXIT

Tabla 72. ***Lista de parámetros para módulos de salida*** *(continuación)*	
Parámetro	**Descripción del elemento**
Nombre de texto	En modalidad de 32 bits: Palabra completa que contiene la dirección de una serie terminada en nulo que contiene el *nombre-texto*completo. Sólo se aplica a FIND. (Sólo lo utiliza LIBEXIT) En modalidad de 64 bits: Palabra doble (8 bytes).
Serie de parámetro de salida de usuario	En modalidad de 32 bits: Palabra completa que contiene la dirección de una matriz de seiselementos, cada elemento de la cual es una estructura que contiene un campo de longitud de 2 bytes seguido de una serie de 64 caracteres que contiene la serie de parámetro de salida. El sexto elemento es la serie MSGEXIT. En modalidad de 64 bits: Palabra doble (8 bytes).
Tipo de línea de código fuente	Media palabra (utilizada únicamente por INEXIT)
Indicador de sentencia	Media palabra (utilizada únicamente por INEXIT)
Número de columna de sentencia	Media palabra (utilizada únicamente por INEXIT)
Reservado	halfword
Extensión de área de trabajo de salida de usuario	Área de trabajo de ocho palabras completas proporcionada por el compilador para que lo utilice el módulo de salida de usuario: • Primera palabra: reservado • Segunda palabra: MSGEXIT
Datos de salida de mensajes	Área de tres medias palabras proporcionada por el compilador: • Primera media palabra: el número de mensaje del mensaje que se va a personalizar • Segunda media palabra: para un mensaje de diagnóstico, la gravedad predeterminada; para un mensaje FIPS, la categoría FIPS como código numérico • Tercera media palabra: la gravedad solicitada por el usuario para los mensajes (-1 para indicar supresión)
Notas: • Al compilar un programa COBOL de 32 bits utilizando **-q32** o **-qaddr(32)**, la salida de usuario también debe compilarse como de 32 bits, y de forma similar para 64 bits, que es el valor predeterminado. • Los desplazamientos de los campos de estructura cambiarán entre 32 bits y 64 bits. En lugar de codificar la salida de usuario para esperar que un campo esté en un desplazamiento determinado, utilice las estructuras C o COBOL proporcionadas a continuación, que coincidirán con el diseño de estructura pasado a la salida de usuario del compilador.	

La estructura de parámetros en COBOL es la siguiente:

```
          01 EXIT-PARM-STRUCT.
             02 EXIT-TYPE           PIC 9(4)   COMP.
             02 EXIT-OPERATION      PIC 9(4)   COMP.
             02 EXIT-RETURNCODE     PIC 9(9)   COMP.
             02 EXIT-RECORDLENGTH   PIC 9(9)   COMP.
             02 EXIT-RECORDADDR     POINTER.
             02 EXIT-WORK-AREA.
                03 INEXIT-WORD      PIC 9(9)   COMP.
                03 LIBEXIT-WORD     PIC 9(9)   COMP.
                03 PRTEXIT-WORD     PIC 9(9)   COMP.
                03 EXTRA-WORD    PIC 9(9)   COMP.
             02 EXIT-TEXT-NAME      POINTER.
             02 EXIT-PARM-STRING    POINTER.
             02 EXIT-LINETYPE       PIC X(2).
             02 EXIT-STMTIND        PIC X(2).
             02 EXIT-STMTCOL        PIC X(2).
             02 EXIT-RESERVED       PIC X(2).
             02 EXIT-XTENDWORKAREA  PIC X(4) OCCURS 8.
             02 EXIT-MESSAGE-DATA.
                03 EXIT-MESSAGE-NUM PIC 9(4)   COMP.
                03 EXIT-DEFAULT-SEV PIC 9(4)   COMP.
                03 EXIT-USER-SEV    PIC S9(4)  COMP.
```

El parámetro Estructura en C es el siguiente:

```
typedef struct ExitString
{
    short length;
    char str[64];
} ExitString;
typedef struct ExitParmStruct
{
    short exitType;
    short exitOperation;
    int exitReturnCode;
    int exitRecordLength;
    void *exitRecord;
    int exitInExitWorkArea;
    int exitLibExitWorkArea;
    int exitPrtExitWorkArea;
    int exitExtraWorkArea;
    char *exitTextName;
    ExitString *exitParmString;
    short exitLineType;
    short exitStmtIndicator;
    short exitStmtColNum;
    short reserved;
    int exitExtendedWorkArea[8];
    struct
    {
        short exitMsgNum;
        short exitDefaultSev;
        short exitUserSev;
    } exitMessageData;
} __attribute__((packed)) ExitParmStruct;
```

Referencias relacionadas
“Área de trabajo de salida de usuario y extensión de área de trabajo” en la página 611

Procesamiento de INEXIT

Si se especifica INEXIT , el compilador carga el módulo de salida (*mod1*) durante la inicialización e invoca el módulo utilizando el código de operación OPEN (código op). A continuación, el módulo puede preparar su fuente para el proceso y volver a pasar el estado de la petición OPEN al compilador. Posteriormente, cada vez que el compilador requiere una sentencia fuente, se invoca el módulo de salida con el código GET op.

A continuación, el módulo de salida devuelve la dirección y la longitud de la siguiente sentencia o la indicación de fin de datos si no existen más sentencias fuente. Cuando se produce el final de los datos, el compilador invoca el módulo de salida con el código de operación CLOSE para que el módulo pueda liberar los recursos relacionados con su entrada.

El compilador utiliza una estructura, pasada por referencia, para comunicarse con el módulo de salida.

Referencias relacionadas
"Lista de parámetros para módulos de salida" en la página 611

Procesamiento de LIBEXIT

Si se especifica LIBEXIT , el compilador carga el módulo de salida (*mod2*) durante la inicialización. El compilador realiza llamadas al módulo para obtener libros de copias siempre que se encuentran sentencias COPY o BASIS .

La primera llamada invoca el módulo con un código de operación OPEN. A continuación, el módulo puede preparar el *nombre-biblioteca* especificado para su proceso. El código OPEN op también se emite la primera vez que se especifica un nuevo *nombre-biblioteca* . El módulo de salida devuelve el estado de la solicitud OPEN al compilador pasando un código de retorno.

Cuando se devuelve la salida invocada con el código OPEN op, el compilador invoca el módulo de salida con un código GET op y el módulo de salida pasa al compilador la longitud y la dirección del registro que se va a copiar del libro de copias activo. La operación GET se repite hasta que el indicador de fin de datos se pasa al compilador.

Cuando se produce el final de los datos, el compilador emite una solicitud CLOSE para que el módulo de salida pueda liberar cualquier recurso relacionado con su entrada.

Sentencias COPY anidadas: Cualquier registro del libro de copias activo puede contener una sentencia COPY . (Sin embargo, las sentencias COPY anidadas no pueden contener la frase REPLACING y una sentencia COPY con la frase REPLACING no puede contener sentencias COPY anidadas.) Cuando se encuentra una sentencia COPY anidada válida, el compilador emite un OPEN y, a continuación, una serie de GET hasta que el compilador recibe un EOD de LIBEXIT.

No se pueden realizar llamadas recursivas a *nombre-texto*. Es decir, un libro de copias sólo se puede nombrar una vez en un conjunto de sentencias COPY anidadas hasta que se alcance el final de los datos para dicho libro de copias.

Cuando el módulo de salida recibe la solicitud OPEN, debe enviar su información de control sobre el libro de copia activo a una pila y, a continuación, completar la acción solicitada por OPEN. El *nombre-texto* recién solicitado (o *nombre-base*) se convierte en el libro de copias activo.

El proceso continúa de la forma normal con una serie de solicitudes GET hasta que el indicador de fin de datos se pasa al compilador.

Nota: La salida de usuario debe continuar pasando un indicador de fin de datos cuando el compilador emite un GET hasta que el compilador emite un mandato CLOSE.

Al final de los datos para el libro de copias activo anidado y cuando el compilador emite una operación CLOSE, el módulo de salida debe sacar su información de control de la pila. La siguiente solicitud del compilador será GET. La salida de usuario debe continuar donde se ha dejado.

A continuación, el compilador invoca el módulo de salida con una solicitud GET, y el módulo de salida debe pasar el mismo registro que se ha pasado anteriormente desde este libro de copias. El compilador verifica que se ha pasado el mismo registro y, a continuación, el proceso continúa con las solicitudes GET hasta que se pasa el indicador de fin de datos.

El compilador utiliza una estructura, pasada por referencia, para comunicarse con el módulo de salida.

Referencias relacionadas
"Lista de parámetros para módulos de salida" en la página 611

Proceso de PRTEXIT

Si se especifica PRTEXIT , el compilador carga el módulo de salida (*mod3*) durante la inicialización. El módulo de salida se utiliza en lugar del archivo SYSPRINT.

El compilador invoca el módulo utilizando el código de operación OPEN (código op). A continuación, el módulo puede preparar su destino de salida para el proceso y volver a pasar el estado de la petición OPEN al compilador. Posteriormente, cada vez que el compilador tiene una línea para imprimir, el módulo de salida se invoca con el código de operación PUT. El compilador proporciona la dirección y la longitud del registro que se va a imprimir, y el módulo de salida devuelve el estado de la solicitud PUT al compilador mediante un código de retorno. El primer byte del registro que se va a imprimir contiene un carácter de control de impresora ANSI.

Antes de que se complete la compilación, el compilador invoca el módulo de salida con el código de operación CLOSE para que el módulo pueda liberar los recursos relacionados con su destino de salida.

El compilador utiliza una estructura, pasada por referencia, para comunicarse con el módulo de salida.

Referencias relacionadas
"Lista de parámetros para módulos de salida" en la página 611

Proceso de MSGEXIT

El módulo MSGEXIT se utiliza para personalizar los mensajes de diagnóstico del compilador y los mensajes FIPS. El módulo puede personalizar un mensaje cambiando su gravedad o suprimiéndolo.

Si el módulo MSGEXIT asigna una gravedad a un mensaje FIPS, el mensaje se convierte en un mensaje de diagnóstico. (El mensaje se muestra en el resumen de los mensajes de diagnóstico en el listado.)

Un resumen MSGEXIT al final del listado del compilador muestra cuántos mensajes se han cambiado en gravedad y cuántos mensajes se han suprimido.

Tabla 73. ***Proceso MSGEXIT***

Acción por compilador	**Acción por módulo de salida**
Carga el módulo de salida (*mod5*) durante la inicialización	
Llama al módulo de salida con un código de operación OPEN (código op)	Opcionalmente procesa *str5* y pasa el estado de la solicitud OPEN al compilador
Llama al módulo de salida con un código de operación MSGSEV (código op) cuando el compilador está a punto de emitir un mensaje de diagnóstico o un mensaje FIPS	Una de las acciones siguientes: • Indica que no hay personalización del mensaje (estableciendo el código de retorno en 0) • Especifica una nueva gravedad para (o supresión de) el mensaje y establece el código de retorno en 4 • Indica que la operación ha fallado (estableciendo el código de retorno en 12)
Llama al módulo de salida con un código de operación CLOSE	Opcionalmente libera almacenamiento y pasa el estado de la solicitud CLOSE al compilador
Suprime el módulo de salida (*mod5*) durante la terminación del compilador	

"Ejemplo: salida de usuario MSGEXIT" en la página 619

Tareas relacionadas
"Personalización de gravedades de mensajes del compilador" en la página 617

Referencias relacionadas
"Lista de parámetros para módulos de salida" en la página 611

Personalización de gravedades de mensajes del compilador

Para cambiar las gravedades de los mensajes del compilador o suprimir los mensajes del compilador (incluidos los mensajes FIPS), realice los pasos que se describen a continuación.

Acerca de esta tarea

1. Codifique y compile un programa COBOL denominado ERRMSG. El programa sólo necesita un párrafo `PROGRAM-ID` , tal como se describe en la tarea relacionada.
2. Revise el listado ERRMSG , que contiene una lista completa de mensajes del compilador con sus números de mensaje, gravedades y texto de mensaje.
3. Decida qué mensajes desea personalizar.

   Para comprender las personalizaciones que son posibles, consulte la referencia relacionada sobre gravedades de mensajes de compilador personalizables.
4. Codifique un módulo MSGEXIT para implementar las personalizaciones.
   a. Verifique que el parámetro de código de operación indica personalización de gravedad de mensaje.
   b. Compruebe los dos valores de entrada en el parámetro de datos de salida de mensajes: el número de mensaje y la gravedad predeterminada para un mensaje de diagnóstico o la categoría FIPS para un mensaje FIPS.

      La categoría FIPS se expresa como código numérico. Para obtener detalles, consulte la referencia relacionada sobre gravedades de mensajes de compilador personalizables.
   c. Para un mensaje que desea personalizar, establezca la gravedad solicitada por el usuario en el parámetro message-exit-data para indicar:
      - Una gravedad de mensaje nueva, codificando gravedad 0, 4, 8 o 12
      - Supresión de mensajes, codificando gravedad -1
   d. Establezca el código de retorno en uno de los valores siguientes:
      - 0, para indicar que el mensaje no se ha personalizado
      - 4, para indicar que el mensaje se ha encontrado y personalizado
      - 12, para indicar que la operación ha fallado y que la compilación debe terminar
5. Compile y enlace el módulo MSGEXIT.

   Asegúrese de que el módulo esté enlazado como biblioteca compartida. Por ejemplo:

   ```
   cob2 -o IGYMGXT -q32 IGYMSGXT.cbl -e IGYMSGXT
   ```
6. Establezca LD_LIBRARY_PATH para que el módulo MSGEXIT esté disponible para el compilador.

   Por ejemplo, si el objeto compartido está en `/u1/cobdev/exits`, utilice este mandato:

   ```
   export LD_LIBRARY_PATH=/u1/cobdev/exits:$LD_LIBRARY_PATH
   ```
7. Vuelva a compilar el programa ERRMSG, pero utilice la opción de compilador `EXIT(MSGEXIT(`*msgmod*`))`, donde *msgmod* es el nombre del módulo MSGEXIT.
8. Revise el listado y compruebe lo siguiente:
   - Gravedades de mensajes actualizadas
   - Mensajes suprimidos (indicados por XX en lugar de la gravedad)
   - Cambios de gravedad no soportados o supresión de mensajes no soportados (indicados por un mensaje de diagnóstico de severity-U y terminación del compilador con el código de retorno 16)

Tareas relacionadas

“Generación de una lista de mensajes del compilador” en la página 246

Referencias relacionadas
“Variables de entorno de ejecución” en la página 234
“Códigos de gravedad para mensajes de diagnóstico del compilador” en la página 245
“Compilador personalizable-gravedades de mensajes” en la página 618
“Efecto de la personalización de mensajes en el código de retorno de compilación” en la página 619
“Manejo de errores para módulos de salida” en la página 623

Compilador personalizable-gravedades de mensajes

Para personalizar las gravedades de los mensajes del compilador, debe comprender las gravedades posibles de los mensajes de diagnóstico del compilador, los niveles o categorías de los mensajes FIPS y las personalizaciones permitidas de las gravedades de los mensajes.

Los posibles códigos de gravedad para los mensajes de diagnóstico del compilador se describen en la referencia relacionada sobre los códigos de gravedad.

En la tabla siguiente se muestran las ocho categorías de mensajes FIPS (FLAGSTD). La categoría de cualquier mensaje FIPS dado se pasa como un código numérico al módulo MSGEXIT. Estos códigos numéricos se muestran en la segunda columna.

Tabla 74. ***Categorías de mensajes FIPS (FLAGSTD)***

Nivel o categoría de FIPS	Código numérico	Descripción
D	81	Nivel 1 de módulo de depuración
E	82	Extensión (IBM)
H	83	Alto nivel
I	84	Nivel intermedio
N	85	Nivel 1 de módulo de segmentación
O	86	Elementos obsoletos
Q	87	Elementos de alto nivel y obsoletos
S	88	Nivel 2 de módulo de segmentación

Los mensajes FIPS tienen una gravedad implícita de cero (gravedad I).

Personalizaciones de gravedad de mensaje permitidas:

Puede cambiar la gravedad de un mensaje de compilador de las siguientes maneras:

- Los mensajes de diagnóstico del compilador severity-I y severity-W, y los mensajes FIPS, se pueden cambiar para que tengan cualquier gravedad de I a S.

  La asignación de una gravedad a un mensaje FIPS convierte el mensaje FIPS en un mensaje de diagnóstico de la gravedad asignada.

  Como ejemplos, puede:

  – Reduzca un aviso de optimizador a gravedad I.
  – No permitir `REDEFINING` un elemento más pequeño con un elemento más grande aumentando la gravedad del mensaje 1154.
  – No permitir `OCCURS DEPENDING ON` complejo cambiando el mensaje 8235 de FIPS de un mensaje FIPS category-E a un mensaje de diagnóstico de compilador severity-S.
- Los mensajes severity-E se pueden elevar a gravedad S, pero no se pueden reducir a gravedad I o W, porque se ha producido una condición de error en el programa.
- Los mensajes severity-S y severity-U no se pueden cambiar para tener una gravedad diferente.

Puede solicitar la supresión de los mensajes del compilador de la forma siguiente:

- Los mensajes I, W y FIPS se pueden suprimir.
- Los mensajes E y S no se pueden suprimir.

Referencias relacionadas
“Códigos de gravedad para mensajes de diagnóstico del compilador” en la página 245
“FLAGSTD” en la página 285
“Efecto de la personalización de mensajes en el código de retorno de compilación” en la página 619

Efecto de la personalización de mensajes en el código de retorno de compilación

Si utiliza un módulo MSGEXIT, el código de retorno final de la compilación de un programa podría verse afectado tal como se describe a continuación.

Si cambia la gravedad de un mensaje, es posible que también se cambie el código de retorno de la compilación. Por ejemplo, si una compilación produce un mensaje de diagnóstico, y es un mensaje de gravedad E, el código de retorno de la compilación sería normalmente 8. Pero si el módulo MSGEXIT cambia la gravedad de ese mensaje a la gravedad S, el código de retorno de la compilación sería 12.

Si suprime un mensaje, el código de retorno de la compilación ya no se ve afectado por la gravedad de dicho mensaje. Por ejemplo, si una compilación produce un mensaje de diagnóstico y es un mensaje de gravedad W, el código de retorno de la compilación sería normalmente 4. Pero si el módulo MSGEXIT suprime ese mensaje, el código de retorno de la compilación sería 0.

Tareas relacionadas
“Personalización de gravedades de mensajes del compilador” en la página 617

Referencias relacionadas
“Códigos de gravedad para mensajes de diagnóstico del compilador” en la página 245

Ejemplo: salida de usuario MSGEXIT

El ejemplo siguiente muestra un módulo de salida de usuario MSGEXIT que cambia las gravedades de los mensajes y suprime los mensajes.

También puede encontrar el código fuente completo para el ejemplo en el subdirectorio `samples` del directorio de instalación de COBOL (normalmente en `/opt/ibm/cobol/1.2.0/samples/msgexit`).

Para obtener sugerencias útiles sobre el uso de un módulo de salida de mensajes, consulte los comentarios dentro del código.

```
*****************************************************************
*  IGYMSGXT - Sample COBOL program for MSGEXIT                  *
*****************************************************************
*  Function:  This is a SAMPLE user exit for the MSGEXIT        *
*             suboption of the EXIT compiler option.  This exit *
*             can be used to customize the severity of or       *
*             suppress compiler diagnostic messages and FIPS    *
*             messages.  This example program includes several  *
*             sample customizations to show how customizations  *
*             are done.  Feel free to change the customizations *
*             as appropriate to meet your requirements.         *
*                                                               *
*---------------------------------------------------------------*
*  COMPILE NOTE: To prepare a compiler user exit in COBOL,      *
*                it should be a shared library module:          *
*   cob2 -o IGYMSGXT -q32 IGYMSGXT.cbl -e IGYMSGXT              *
*                                                               *
*  USAGE NOTE: The compiler needs to have access to IGYMSGXT at *
*              compile time, so set LD_LIBRARY_PATH accordingly:*
*                                                               *
*   EX:       export LD_LIBRARY_PATH=/u1/cobdev/exits:          *
*             $LD_LIBRARY_PATH                                  *
*                                                               *
*               (This assumes the shared object is in           *
*                 /u1/cobdev/exits )                            *
*                                                               *
*****************************************************************
```

```
*****************************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  IGYMSGXT.
 DATA DIVISION.

   WORKING-STORAGE SECTION.

*****************************************************************
*                                                               *
*   Local variables.                                            *
*                                                               *
*****************************************************************

    77 EXIT-TYPEN            PIC 9(4).
    77 EXIT-DEFAULT-SEV-FIPS PIC X.

*****************************************************************
*                                                               *
*   Definition of the User-Exit Parameter List, which is        *
*   passed from the COBOL compiler to the user-exit module      *
*                                                               *
*****************************************************************

   LINKAGE SECTION.
    01 UXPARM.
       02 EXIT-TYPE        PIC 9(4)   COMP.
       02 EXIT-OPERATION   PIC 9(4)   COMP.
       02 EXIT-RETURNCODE  PIC 9(9)   COMP.
       02 EXIT-DATALENGTH  PIC 9(9)   COMP.
       02 EXIT-DATA        POINTER.
       02 EXIT-WORK-AREA.
          03 EXIT-WORK-AREA-PTR  OCCURS 4  POINTER.
       02 EXIT-TEXT-NAME   POINTER.
       02 EXIT-PARMS       POINTER.
       02 EXIT-LINFO       PIC X(8).
       02 EXIT-X-WORK-AREA PIC X(4) OCCURS 8.
       02 EXIT-MESSAGE-PARMS.
          03 EXIT-MESSAGE-NUM PIC 9(4)   COMP.
          03 EXIT-DEFAULT-SEV PIC 9(4)   COMP.
          03 EXIT-USER-SEV    PIC S9(4)  COMP.

    01 EXIT-STRINGS.
       02 EXIT-STRING OCCURS 6.
          03 EXIT-STR-LEN PIC 9(4)   COMP.
          03 EXIT-STR-TXT PIC X(64).

*****************************************************************
*                                                               *
*  Begin PROCEDURE DIVISION                                     *
*                                                               *
*  Invoke the section to handle the exit.                       *
*                                                               *
*****************************************************************

 Procedure Division Using UXPARM.

     Set Address of EXIT-STRINGS to EXIT-PARMS

     COMPUTE EXIT-RETURNCODE = 0

     Evaluate
TRUE

*****************************************************************
* Handle a bad invocation of this exit by the compiler.         *
* This could happen if this routine was used for one of the     *
* other EXITs, such as INEXIT, PRTEXIT or LIBEXIT.              *
*****************************************************************
       When EXIT-TYPE Not = 6
       Move EXIT-TYPE   to  EXIT-TYPEN
         Display '**** Invalid exit routine identifier'
         Display '**** EXIT TYPE =  '  EXIT-TYPE
         Compute EXIT-RETURNCODE = 16

*****************************************************************
* Handle the OPEN call to this exit by the compiler             *
*        Display the exit string (labeled 'str5' in the syntax  *
*        diagram in the COBOL for Linux Programming Guide) from   *
*        the EXIT(MSGEXIT('str5',mod5)) option specification.   *
*        (Note that str5 is placed in element 6 of the array of *
*        user exit parameter strings.)                          *
*****************************************************************
```

```
       When EXIT-OPERATION = 0
*        Display 'Opening MSGEXIT'
*        If EXIT-STR-LEN(6) Not Zero Then
*          Display ' str5 len = ' EXIT-STR-LEN(6)
*          Display ' str5 = ' EXIT-STR-TXT(6)(1:EXIT-STR-LEN(6))
*        End-If
         Continue

*****************************************************************
* Handle the CLOSE call to this exit by the compiler            *
*        NOTE: Unlike the z/OS MSGEXIT, you must not use        *
*             STOP RUN here.  On Linux, use GOBACK.               *
*****************************************************************
       When EXIT-OPERATION = 1
*        Display 'Closing MSGEXIT'
         Goback

*****************************************************************
* Handle the customize message severity call to this exit       *
*        Display information about every customized severity.   *
*****************************************************************
       When EXIT-OPERATION = 5
*        Display 'MSGEXIT called with MSGSEV'
         If EXIT-MESSAGE-NUM < 8000 Then
           Perform Error-Messages-Severity
         Else
           Perform FIPS-Messages-Severity
         End-If

*        If EXIT-RETURNCODE = 4 Then
*          Display '>>>> Customizing message ' EXIT-MESSAGE-NUM
*                  ' with new severity ' EXIT-USER-SEV '  <<<<'
*          If EXIT-MESSAGE-NUM > 8000 Then
*            Display 'FIPS sev =' EXIT-DEFAULT-SEV-FIPS '<<<<'
*          End-If
*        End-If

*****************************************************************
* Handle a bad invocation of this exit by the compiler          *
* The compiler should not invoke this exit with EXIT-TYPE = 6   *
* and an opcode other than 0, 1, or 5.  This should not happen  *
* and IBM service should be contacted if it does.               *
*****************************************************************
       When Other
         Display '**** Invalid MSGEXIT routine operation '
         Display '**** EXIT OPCODE =  '  EXIT-OPERATION
         Compute EXIT-RETURNCODE = 16

     End-Evaluate


Goback.


*****************************************************************
*    ERROR MESSAGE   PROCESSOR                                  *
*****************************************************************
 Error-Messages-Severity.

*    Assume message severity will be customized...
     COMPUTE EXIT-RETURNCODE = 4

     Evaluate EXIT-MESSAGE-NUM

*****************************************************************
*      Change severity of message 1154(W) to 12 ("S")
*      This is the case of redefining a large item
*      with a smaller item, IBM Req # MR0904063236
*****************************************************************
       When(1154)
         COMPUTE EXIT-USER-SEV = 12

*****************************************************************
*      Message severity Not customized
*****************************************************************
       When Other
         COMPUTE EXIT-RETURNCODE = 0

     End-Evaluate
     .
*****************************************************************
*    FIPS MESSAGE   PROCESSOR                                   *
```

```
*****************************************************************
 Fips-Messages-Severity.

*    Assume message severity will be customized...
     COMPUTE EXIT-RETURNCODE = 4

*    Convert numeric 'category' to character
     EVALUATE EXIT-DEFAULT-SEV
       When 81
         MOVE 'D' To EXIT-DEFAULT-SEV-FIPS
       When 82
         MOVE 'E' To EXIT-DEFAULT-SEV-FIPS
       When 83
         MOVE 'H' To EXIT-DEFAULT-SEV-FIPS
       When 84
         MOVE 'I' To EXIT-DEFAULT-SEV-FIPS
       When 85
         MOVE 'N' To EXIT-DEFAULT-SEV-FIPS
       When 86
         MOVE 'O' To EXIT-DEFAULT-SEV-FIPS
       When 87
         MOVE 'Q' To EXIT-DEFAULT-SEV-FIPS
       When 88
         MOVE 'S' To EXIT-DEFAULT-SEV-FIPS
       When Other
         Continue
     End-Evaluate

*****************************************************************
*  Examples of using FIPS category to force coding
*  restrictions.  These are not recommendations!
*****************************************************************
*      Change severity of all OBSOLETE item FIPS
*       messages to 'S'
*****************************************************************
*    If EXIT-DEFAULT-SEV-FIPS = 'O' Then
*      DISPLAY ">>>> Default customizing FIPS category "
*        EXIT-DEFAULT-SEV-FIPS " msg " EXIT-MESSAGE-NUM "<<<<"
*      COMPUTE EXIT-USER-SEV = 12
*    End-If

     Evaluate EXIT-MESSAGE-NUM
*****************************************************************
*      Change severity of message 8062(O) to 8 ("E")
*        8062 = GO TO without proc name
*****************************************************************
       When(8062)
*        DISPLAY ">>>> Customizing message 8062 with 8 <<<<"
*        DISPLAY 'FIPS sev =' EXIT-DEFAULT-SEV-FIPS '='
         COMPUTE EXIT-USER-SEV = 8

*****************************************************************
*      Change severity of message 8193(E) to 0("I")
*        8193 = GOBACK
*****************************************************************
       When(8193)
*        DISPLAY ">>>> Customizing message 8193 with 0 <<<<"
*        DISPLAY 'FIPS sev =' EXIT-DEFAULT-SEV-FIPS '='
         COMPUTE EXIT-USER-SEV = 0

*****************************************************************
*      Change severity of message 8235(E) to 8 (Error)
*      to disalllow Complex Occurs Depending On
*        8235 = Complex Occurs Depending On
*****************************************************************
       When(8235)
*        DISPLAY ">>>> Customizing message 8235 with 8 <<<<"
*        DISPLAY 'FIPS sev =' EXIT-DEFAULT-SEV-FIPS '='
         COMPUTE EXIT-USER-SEV = 08

*****************************************************************
*      Change severity of message 8270(O) to -1 (Suppress)
*        8270 = SERVICE LABEL
*****************************************************************
       When(8270)
*        DISPLAY ">>>> Customizing message 8270 with -1 <<<<"
*        DISPLAY 'FIPS sev =' EXIT-DEFAULT-SEV-FIPS '='
         COMPUTE EXIT-USER-SEV = -1

*****************************************************************
*      Message severity Not customized
*****************************************************************
```

```
       When Other
*        For the default set 'O' to 'S' case...
*        If EXIT-USER-SEV = 12 Then
*          COMPUTE EXIT-RETURNCODE = 4
*        Else
           COMPUTE EXIT-RETURNCODE = 0
*        End-If

     End-Evaluate
     .
 END PROGRAM IGYMSGXT.
```

Manejo de errores para módulos de salida

Las condiciones descritas a continuación pueden producirse durante el proceso de las salidas de usuario.

Anomalía de carga de salida:

El mensaje IGYSI5207-U se graba en el operador si falla una solicitud LOAD para cualquiera de las salidas de usuario:

```
An error occurred while attempting to load user exit exit-name.
```

Anomalía de apertura de salida:

El mensaje IGYSI5208-U se graba en el operador si falla una solicitud OPEN para cualquiera de las salidas de usuario:

```
An error occurred while attempting to open user exit exit-name.
```

Anomalía de PUT de PRTEXIT:

- El mensaje IGYSI5203-U se escribe en el listado:

  ```
  A PUT request to the PRTEXIT user exit failed with return code nn.
  ```

- El mensaje IGYSI5217-U se escribe en el operador:

  ```
  An error occurred in PRTEXIT user exit exit-name. Compiler terminated.
  ```

Anomalías de SYSIN GET:

Es posible que se graben los mensajes siguientes en el listado:

- IGYSI5204-U:

  ```
  The record address was not set by the exit-name user exit.
  ```

- IGYSI5205-U:

  ```
  A GET request from the INEXIT user exit failed with return code nn.
  ```

- IGYSI5206-U:

  ```
  The record length was not set by the exit-name user exit.
  ```

Anomalías de MSGEXIT:

Anomalía de personalización: El mensaje IGY*PP*5293-U se graba en el listado si se intenta un cambio de gravedad no soportado o una supresión de mensajes no soportada:

```
MSGEXIT user exit exit-name specified a message severity customization that is
not supported. The message number, default severity, and user-specified severity
were: mm, ds, us. Change MSGEXIT user exit exit-name to correct this error.
```

Anomalía general: El mensaje IGY*PP*5064-U se graba en el listado si el módulo MSGEXIT establece el código de retorno en un valor distinto de cero distinto de 4:

```
A call to the MSGEXIT user exit routine exit-name failed with return code nn.
```

En los mensajes MSGEXIT, los dos caracteres *PP* indican la fase del compilador que ha emitido el mensaje que ha dado como resultado una llamada al módulo MSGEXIT.

Tareas relacionadas
“Personalización de gravedades de mensajes del compilador” en la página 617

Apéndice F. Mensajes de tiempo de ejecución

Los mensajes para COBOL para Linux contienen un prefijo de mensaje, número de mensaje, código de gravedad y texto descriptivo.

El prefijo de mensaje es siempre IWZ. El código de gravedad es I (información), W (aviso), S (grave) o C (crítico). El texto del mensaje proporciona una breve explicación de la condición.

```
IWZ2519S  The seconds value in a CEEISEC call was not recognized.
```

En el mensaje de ejemplo anterior:

- El prefijo del mensaje es IWZ.
- El número de mensaje es 2519.
- El código de gravedad es S.
- El texto del mensaje es "No se ha reconocido el valor de segundos en una llamada CEEISEC".

Los mensajes de servicios invocables de Fecha y hora también contienen un código de comentarios simbólico, que representa los primeros 8 bytes de una señal de condición de 12 bytes. Puede pensar en el código de comentarios simbólicos como el apodo de una condición. Los mensajes de servicios invocables contienen un número de mensaje de cuatro dígitos.

Al ejecutar la aplicación desde la línea de mandatos, puede capturar cualquier mensaje de tiempo de ejecución redirigiendo stdout y stderr a un archivo. Por ejemplo:

program-name program-arguments >combined-output-file 2>&1

El ejemplo siguiente muestra cómo escribir la salida en archivos separados:

program-name program-arguments >output-file 2>*error-file*

Tabla 75. ***Mensajes de tiempo de ejecución***

Número de mensaje	**texto de mensaje**
“IWZ006S” en la página 633	La referencia a la tabla *nombre-tabla* por número de verbo *número-verbo* en la línea *número-línea* se refiere a un área fuera de la región de la tabla.
“IWZ007S” en la página 633	La referencia al grupo de longitud variable *nombre-grupo* por el número de verbo *número-verbo* en la línea *número-línea* se refiere a un área fuera de la longitud máxima definida del grupo.
“IWZ012I” en la página 633	Se ha producido una terminación de unidad de ejecución no válida mientras se estaba ejecutando la clasificación o fusión.
“IWZ013S” en la página 633	Se ha solicitado la clasificación o fusión mientras la clasificación o fusión se está ejecutando en una hebra diferente.
“IWZ026W” en la página 634	Nunca se ha hecho referencia al registro especial SORT-RETURN , pero el contenido actual ha indicado que la operación de clasificación o fusión en el programa *nombre-programa* en el número de línea *número-línea* no ha sido satisfactoria. El código de retorno de clasificación o fusión era *código de retorno*.
“IWZ029S” en la página 634	Argument-1 para la función *nombre-función* en el programa *nombre-programa* en la línea *número-línea* era menor que cero.
“IWZ030S” en la página 634	Argument-2 para la función *nombre-función* en el programa *nombre-programa* en la línea *número-línea* no era un entero positivo.

*Tabla 75. **Mensajes de tiempo de ejecución** (continuación)*	
"IWZ036W" en la página 634	Se ha producido un truncamiento de posiciones de dígitos de orden superior en el programa *nombre-programa* en el número de línea *número-línea*.
"IWZ037I" en la página 635	El flujo de control en el programa *nombre-programa* ha seguido más allá de la última línea del programa. Control devuelto al interlocutor del programa *nombre-programa*.
"IWZ038S" en la página 635	Se ha encontrado un valor de longitud de modificación de referencia de *valor-modificación-referencia* en la línea *número-línea* que no era igual a 1 en una referencia al elemento de datos *elemento-datos*.
"IWZ039S" en la página 635	Se ha detectado un signo de sobreperforación no válido.
"IWZ040S" en la página 636	Se ha detectado un signo separado no válido.
"IWZ048W" en la página 636	Se ha elevado una base negativa a una potencia fraccional en una expresión de exponenciación. Se utilizó el valor absoluto de la base.
"IWZ049W" en la página 636	Se ha elevado una base cero a una potencia cero en una expresión de exponenciación. El resultado se ha establecido en uno.
"IWZ050S" en la página 636	Se ha elevado una base cero a una potencia negativa en una expresión de exponenciación.
"IWZ051S" en la página 637	No quedan dígitos significativos en una operación de exponenciación de punto fijo en el programa *nombre-programa* debido a que se han especificado demasiadas posiciones decimales en los operandos o receptores.
"IWZ053S" en la página 637	Se ha producido un desbordamiento en la conversión a coma flotante.
"IWZ054S" en la página 637	Se ha producido una excepción de coma flotante.
"IWZ055W" en la página 637	Se ha producido un subdesbordamiento en la conversión a coma flotante. El resultado se ha establecido en cero.
"IWZ058S" en la página 638	Se ha producido un desbordamiento de exponente.
"IWZ059W" en la página 638	Se ha truncado un exponente con más de nueve dígitos.
"IWZ060W" en la página 638	Se ha producido el truncamiento de posiciones de dígitos de orden superior.
"IWZ061S" en la página 638	Se ha producido una división por cero.
"IWZ063S" en la página 639	Se ha detectado un signo no válido en un campo de envío editado numérico en *nombre-programa* en el número de línea *número-línea*.
"IWZ064S" en la página 639	Se ha intentado una llamada recursiva al programa activo *nombre-programa* en la unidad de compilación *unidad-compilación* .
"IWZ065I" en la página 639	Se ha intentado una operación CANCEL del programa activo *nombre-programa* en la unidad de compilación *unidad-compilación* .
"IWZ066S" en la página 639	La longitud del registro de datos externo *registro-datos* no coincidía con la longitud existente del registro.

Tabla 75. ***Mensajes de tiempo de ejecución*** *(continuación)*	
“IWZ071S” en la página 639	La referencia de tabla con subíndice ALL a la tabla *nombre-tabla* por número de verbo *número-verbo* en línea *número-línea* tenía un subíndice ALL especificado para una dimensión OCCURS DEPENDING ON, y el objeto era menor o igual que 0.
“IWZ072S” en la página 640	Un valor de posición inicial de modificación de referencia de *valor-modificación-referencia* en la línea *número-línea* hacía referencia a un área fuera de la región del elemento de datos *elemento-datos*.
“IWZ073S” en la página 640	Se ha encontrado un valor de longitud de modificación de referencia no positiva de *valor-modificación-referencia* en la línea *número-línea* en una referencia al elemento de datos *elemento-datos*.
“IWZ074S” en la página 640	Un valor de posición inicial de modificación de referencia de *valor-modificación-referencia* y un valor de longitud de *longitud* en línea *número-línea* ha hecho que la referencia se realice más allá del carácter situado más a la derecha del elemento de datos *elemento de datos*.
“IWZ075S” en la página 641	Se han encontrado incoherencias en el archivo EXTERNAL *nombre-archivo* en el programa *nombre-programa*. Los atributos de archivo siguientes no coinciden con los del archivo externo establecido: *attribute-1 attribute-2 attribute-3 attribute-4 attribute-5 attribute-6 attribute-7*.
“IWZ076W” en la página 641	El número de caracteres en el nombre de datos INSPECT SUSTITUT CHARACTERS BY no era igual a uno. Se ha utilizado el primer carácter.
“IWZ077W” en la página 641	Las longitudes de los elementos de datos INSPECT no eran iguales. Se ha utilizado la longitud más corta.
“IWZ078S” en la página 641	TODA la referencia de tabla con subíndice a la tabla *nombre-tabla* por número de verbo *número-verbo* en la línea *número-línea* superará el límite superior de la tabla.
“IWZ096C” en la página 642	Las variantes de mensaje incluyen: • La llamada dinámica del programa *nombre-programa* ha fallado. Una carga del módulo *nombre-módulo* ha fallado con un código de error de *códo-error*. • La llamada dinámica del programa *nombre-programa* ha fallado. Una carga del módulo *nombre-módulo* ha fallado con un código de retorno de *códo-retorno*. • La llamada dinámica del programa *nombre-programa* ha fallado. Recursos insuficientes. • La llamada dinámica del programa *nombre-programa* ha fallado. COBPATH no se ha encontrado en el entorno. • La llamada dinámica del programa *nombre-programa* ha fallado. No se ha encontrado la entrada *nombre-entrada* . • La llamada dinámica ha fallado. El nombre del programa de destino no contiene ningún carácter válido. • La llamada dinámica del programa *nombre-programa* ha fallado. El módulo de carga *módulo de carga* no se ha podido encontrar en los directorios identificados en la variable de entorno COBPATH.
“IWZ097S” en la página 642	Argument-1 para la función *nombre-función* no contenía dígitos.
“IWZ100S” en la página 642	Argument-1 para la función *nombre-función* era menor o igual que -1.

Tabla 75. ***Mensajes de tiempo de ejecución*** *(continuación)*	
"IWZ103S" en la página 643	Argument-1 para la función *nombre-función* era menor que cero o mayor que 99.
"IWZ104S" en la página 643	Argument-1 para la función *nombre-función* era menor que cero o mayor que 99999.
"IWZ105S" en la página 643	Argument-1 para la función *nombre-función* era menor que cero o mayor que 999999.
"IWZ151S" en la página 643	Argument-1 para la función *nombre-función* contenía más de 18 dígitos.
"IWZ152S" en la página 643	Se ha encontrado un carácter no válido *carácter* en la columna *número-columna* en el argument-1 para la función *nombre-función*.
"IWZ155S" en la página 643	Se ha encontrado un carácter no válido *carácter* en la columna *número-columna* en el argument-2 para la función *nombre-función*.
"IWZ156S" en la página 644	Argument-1 para la función *nombre-función* era menor que cero o mayor que 28.
"IWZ157S" en la página 644	La longitud de Argument-1 para la función *nombre-función* no era igual a 1.
"IWZ158S" en la página 644	Argument-1 para la función *nombre-función* era menor que cero o mayor que 29.
"IWZ159S" en la página 644	Argument-1 para la función *nombre-función* era menor que 1 o mayor que 3067671.
"IWZ160S" en la página 644	Argument-1 para la función *nombre-función* era menor que 16010101 o mayor que 99991231.
"IWZ161S" en la página 645	Argument-1 para la función *nombre-función* era menor que 1601001 o mayor que 999999365.
"IWZ162S" en la página 645	Argument-1 para la función *nombre-función* era menor que 1 o mayor que el número de posiciones en la secuencia de clasificación del programa.
"IWZ163S" en la página 645	Argument-1 para la función *nombre-función* era menor que cero.
"IWZ165S" en la página 645	Un valor de posición inicial de modificación de referencia de *valor-posición-inicio* en la línea *número de línea* hacía referencia a un área fuera de la región del resultado de la función de *resultado-función*.
"IWZ166S" en la página 645	Se ha encontrado un valor de longitud de modificación de referencia no positiva de *longitud* en la línea *número-línea* en una referencia al resultado de función de *resultado-función*.
"IWZ167S" en la página 646	Un valor de posición inicial de modificación de referencia de *posición inicial* y un valor de longitud de *longitud* en línea *número-línea* han hecho que la referencia se realice más allá del carácter situado más a la derecha del resultado de la función de *resultado-función*.
"IWZ168W" en la página 646	SYSPUNCH/SYSPCH tomará por omisión el dispositivo de salida lógica del sistema. No se ha establecido la variable de entorno correspondiente.
"IWZ169S" en la página 646	Tipo de dispositivo desconocido para la sentencia DISPLAY.
"IWZ170S" en la página 646	Tipo de datos no permitido para el operando DISPLAY.

Tabla 75. ***Mensajes de tiempo de ejecución*** *(continuación)*	
“IWZ171I” en la página 647	*nombre-serie* no es una opción de tiempo de ejecución válida.
“IWZ172I” en la página 647	La serie *nombre-serie* no es una subopción válida de la opción de tiempo de ejecución *nombre-opción*.
“IWZ173I” en la página 647	La serie de subopción *nombre-serie* de la opción de tiempo de ejecución *nombre-opción* debe tener *número* caracteres de longitud. Se utilizará el valor predeterminado.
“IWZ174I” en la página 647	La serie de subopción *nombre-serie* de la opción de tiempo de ejecución *nombre-opción* contiene uno o más caracteres no válidos. Se utilizará el valor predeterminado.
“IWZ175S” en la página 647	No hay soporte para la rutina *nombre-rutina* en este sistema.
“IWZ176S” en la página 648	Argument-1 para la función *nombre-función* era mayor que *valor-decimal*.
“IWZ177S” en la página 648	Argument-2 para la función *nombre-función* era igual a *valor-decimal*.
“IWZ178S” en la página 648	Argument-1 para la función *nombre-función* era menor o igual que *valor-decimal*.
“IWZ179S” en la página 648	Argument-1 para la función *nombre-función* era menor que *valor-decimal*.
“IWZ180S” en la página 648	Argument-1 para la función *nombre-función* no era un entero.
“IWZ181I” en la página 649	Se ha encontrado un carácter no válido en la serie numérica *serie* de la opción de tiempo de ejecución *nombre-opción*. Se utilizará el valor predeterminado.
“IWZ182I” en la página 649	El número *número* de la opción de tiempo de ejecución *nombre-opción* ha superado el rango de *rango-mínimo* a *rango-máximo*. Se utilizará el valor predeterminado.
“IWZ183S” en la página 649	El nombre de función en _IWZCOBOLInit ha devuelto un retorno.
“IWZ200S” en la página 649	Se ha detectado un error durante la *operación de E/S* para el archivo *nombre-archivo*. El estado del archivo es: *file-status*.
“IWZ200S” en la página 649	STOP o ACCEPT han fallado con un error de E/S, *código de error*. La unidad de ejecución se cancela.
“IWZ201C” en la página 650	
“IWZ203S” en la página 650	La página de códigos en vigor no es una página de códigos DBCS.
“IWZ204S” en la página 651	Se ha producido un error durante la conversión de DBCS ASCII a DBCS EBCDIC.
“IWZ221S” en la página 651	El conversor ICU para la página de códigos , *valor de página de códigos,* no se puede abrir. El código de error es *valor de código de error*.
“IWZ222S” en la página 651	La conversión de datos a través de ICU ha fallado con el código de error *valor de código de error*.

Tabla 75. ***Mensajes de tiempo de ejecución*** *(continuación)*	
"IWZ223W" en la página 651	El cierre del conversor ICU ha fallado con el código de error *valor de código de error*.
"IWZ224S" en la página 651	El colador ICU para el valor de entorno local, *valor de entorno local*, no se puede abrir. El código de error es *valor de código de error*.
"IWZ225S" en la página 652	La función de correlación de casos Unicode que utiliza ICU ha fallado con el código de error *valor de código de error*. El entorno local en vigor es *valor de entorno local*.
"IWZ230W" en la página 652	La tabla de conversión para la página de códigosactual, *ASCII codeset-id*, a la página de códigosEBCDIC, *EBCDIC codeset-id*, no está disponible. Se utilizará la tabla de conversión ASCII a EBCDIC predeterminada.
"IWZ230W" en la página 652	La página de códigos EBCDIC especificada, *página de códigos EBCDIC*, no es coherente con el entorno local *entorno local*, pero se utilizará según se solicite.
"IWZ230W" en la página 652	La página de códigos EBCDIC especificada, *Página de códigos EBCDIC*, no está soportada. Se utilizará la página de códigos EBCDIC predeterminada, *página de códigos EBCDIC*.
"IWZ230S" en la página 653	La tabla de conversión EBCDIC no se puede abrir.
"IWZ230S" en la página 653	No se puede crear la tabla de conversión EBCDIC.
"IWZ230S" en la página 653	El programa principal se ha compilado con el distintivo `-host` y la opción `CHAR(NATIVE)` , que no son compatibles.
"IWZ231S" en la página 653	La consulta del valor de entorno local actual ha fallado.
"IWZ232W" en la página 653	Las variantes de mensaje incluyen: • Se ha producido un error durante la conversión del elemento de datos *nombre-datos* a EBCDIC en el programa *nombre-programa* en el número de línea *valor-decimal*. • Se ha producido un error durante la conversión del elemento de datos *nombre-datos* a ASCII en el programa *nombre-programa* en el número de línea *valor-decimal*. • Se ha producido un error durante la conversión a EBCDIC para el elemento de datos *nombre-datos* en el programa *nombre-programa* en el número de línea *valor-decimal*. • Se ha producido un error durante la conversión a ASCII para el elemento de datos *nombre-datos* en el programa *nombre-programa* en el número de línea *valor-decimal*. • Se ha producido un error durante la conversión de ASCII a EBCDIC en el programa *nombre-programa* en el número de línea *valor-decimal*. • Se ha producido un error durante la conversión de EBCDIC a ASCII en el programa *nombre-programa* en el número de línea *valor-decimal*.
"IWZ240S" en la página 654	El año base para el programa *nombre-programa* estaba fuera del rango válido de 1900 a 1999. El valor de ventana deslizante *valor-ventana* ha dado como resultado un año base de *año-base*.
"IWZ241S" en la página 654	El año actual estaba fuera de la ventana de 100 años, de *inicio de año* a *fin de año*, para el programa *nombre-programa*.

Tabla 75. ***Mensajes de tiempo de ejecución*** *(continuación)*	
"IWZ242S" en la página 655	Se ha producido un intento no válido de iniciar una sentencia XML PARSE.
"IWZ243S" en la página 655	Se ha producido un intento no válido de finalizar una sentencia XML PARSE.
"IWZ813S" en la página 655	No había suficiente almacenamiento disponible para satisfacer una solicitud de obtención de almacenamiento.
"IWZ901S" en la página 655	Las variantes de mensaje incluyen: • Salidas de programa debido a un error grave o crítico. • Salidas de programa: se han producido más de ERRCOUNT errores.
"IWZ902S" en la página 656	El sistema ha detectado una excepción Decimal-divide .
"IWZ903S" en la página 656	El sistema ha detectado una excepción de datos.
"IWZ907S" en la página 656	Las variantes de mensaje incluyen: • Almacenamiento insuficiente. • Almacenamiento insuficiente. No se pueden obtener *number-bytes* bytes de espacio para el *almacenamiento*.
"IWZ993W" en la página 656	Almacenamiento insuficiente. No se puede encontrar espacio para el mensaje *message-number*.
"IWZ994W" en la página 657	No se puede encontrar el mensaje *message-number* en *message-catalog*.
"IWZ995C" en la página 657	Las variantes de mensaje incluyen: • Se ha recibido la señal *Excepción del sistema* al ejecutar la rutina *nombre-rutina* en el desplazamiento 0x*valor-desplazamiento*. • Se ha recibido la señal de *excepción del sistema* al ejecutar el código en la ubicación 0x*valor-desplazamiento*. • Se ha recibido la señal *Excepción del sistema* . No se ha podido determinar la ubicación.
"IWZ2502S" en la página 657	El UTC/GMT no estaba disponible en el sistema.
"IWZ2503S" en la página 657	El desplazamiento de UTC/GMT a la hora local no estaba disponible desde el sistema.
"IWZ2505S" en la página 658	El valor de input_seconds en una llamada a CEEDATM o CEESECI no estaba dentro del rango soportado.
"IWZ2506S" en la página 658	Se ha utilizado una era (< JJJ>, < CCCC> o < CCCCCCCCCC>) en una serie de imagen pasada a CEEDATM, pero el valor de número de segundos de entrada no estaba dentro del rango soportado. No se pudo determinar la era.
"IWZ2507S" en la página 658	No se pasaron datos suficientes a CEEDAYS o CEESECS. El valor Lilian no se ha calculado.
"IWZ2508S" en la página 658	El valor de fecha pasado a CEEDAYS o CEESECS no era válido.

*Tabla 75. **Mensajes de tiempo de ejecución** (continuación)*	
“IWZ2509S” en la página 659	La era pasada a CEEDAYS o CEESECS no fue reconocida.
“IWZ2510S” en la página 659	No se ha reconocido el valor de horas en una llamada a CEEISEC o CEESECS.
“IWZ2511S” en la página 659	El parámetro de día pasado en una llamada CEEISEC no era válido para el año y el mes especificados.
“IWZ2512S” en la página 660	El valor de fecha Lilian pasado en una llamada a CEEDATE o CEEDYWK no estaba dentro del rango soportado.
“IWZ2513S” en la página 660	La fecha de entrada pasada en una llamada CEEISEC, CEEDAYS o CEESECS no estaba dentro del rango soportado.
“IWZ2514S” en la página 660	El valor de año pasado en una llamada CEEISEC no estaba dentro del rango soportado.
“IWZ2515S” en la página 660	No se ha reconocido el valor de milisegundos en una llamada CEEISEC.
“IWZ2516S” en la página 661	No se ha reconocido el valor de minutos en una llamada CEEISEC.
“IWZ2517S” en la página 661	No se ha reconocido el valor de mes en una llamada CEEISEC.
“IWZ2518S” en la página 661	Se ha especificado una serie de imagen no válida en una llamada a un servicio de fecha/hora.
“IWZ2519S” en la página 661	No se ha reconocido el valor de segundos en una llamada CEEISEC.
“IWZ2520S” en la página 662	CEEDAYS ha detectado datos no numéricos en un campo numérico, o la serie de fecha no coincidía con la serie de imagen.
“IWZ2521S” en la página 662	El valor de año dentro de era < JJJJ>, < CCCC> o < CCCCCCCCCC> pasado a CEEDAYS o CEESECS era cero.
“IWZ2522S” en la página 662	Se ha utilizado una era (< JJJJ>, < CCCC> o < CCCCCCCCCC>) en una serie de imagen pasada a CEEDATE, pero el valor de fecha de Lilian no estaba dentro del rango soportado. No se pudo determinar la era.
“IWZ2525S” en la página 662	CEESECS ha detectado datos no numéricos en un campo numérico, o la serie de indicación de fecha y hora no coincidía con la serie de imagen.
“IWZ2526S” en la página 663	La serie de fecha devuelta por CEEDATE se ha truncado.
“IWZ2527S” en la página 663	La serie de indicación de fecha y hora devuelta por CEEDATM se ha truncado.
“IWZ2531S” en la página 663	La hora local no estaba disponible en el sistema.
“IWZ2533S” en la página 663	El valor pasado a CEESCEN no estaba entre 0 y 100.
“IWZ2534W” en la página 664	Se ha especificado una anchura de campo insuficiente para un nombre de mes o día de la semana en una llamada a CEEDATE o CEEDATM. La salida se ha establecido en blancos.

IWZ006S

La referencia a la tabla *nombre-tabla* por número de verbo *número-verbo* en línea *número-línea* se refiere a un área fuera de la región de la tabla.

Explicación: Cuando la opción SSRANGE está en vigor, este mensaje se emite para indicar que una tabla de longitud fija se ha subscripto de una forma que excede el tamaño definido de la tabla o, para tablas de longitud variable, el tamaño máximo de la tabla.

La comprobación de rango se ha realizado en el compuesto de los subíndices y ha dado como resultado una dirección fuera de la región de la tabla. Para tablas de longitud variable, la dirección está fuera de la región de la tabla definida cuando todos los objetos OCCURS DEPENDING ON están en sus valores máximos; no se tiene en cuenta el valor actual del objeto ODO. La comprobación no se ha realizado en subíndices individuales.

Respuesta del programador: Asegúrese de que el valor de los subíndices literales y el valor de los subíndices de variable tal como se evalúan en tiempo de ejecución no sobrepasen las dimensiones de subíndice para los datos de subíndice en la sentencia anómala.

Acción del sistema: La aplicación ha terminado.

IWZ007S

La referencia al grupo de longitud variable *nombre-grupo* por número de verbo *número-verbo* en línea *número-línea* se refiere a un área fuera de la longitud máxima definida del grupo.

Explicación: Cuando la opción SSRANGE está en vigor, este mensaje se emite para indicar que un grupo de longitud variable generado por OCCURS DEPENDING ON tiene una longitud inferior a cero o superior a los límites definidos en las cláusulas OCCURS DEPENDING ON.

La comprobación de rango se ha realizado en la longitud compuesta del grupo y no en los objetos OCCURS DEPENDING ON individuales.

Respuesta del programador: Asegúrese de que los objetos OCCURS DEPENDING ON evaluados en tiempo de ejecución no excedan el número máximo de apariciones de la dimensión para las tablas dentro del elemento de grupo al que se hace referencia.

Acción del sistema: La aplicación ha terminado.

IWZ012I

Se ha producido una terminación de unidad de ejecución no válida mientras se estaba ejecutando la clasificación o fusión.

Explicación: Una clasificación o fusión iniciada por un programa COBOL estaba en curso y se ha intentado una de las siguientes acciones:

1. Se ha emitido STOP RUN.
2. Se ha emitido un GOBACK o un EXIT PROGRAM dentro del procedimiento de entrada o del procedimiento de salida del programa COBOL que ha iniciado la clasificación o fusión. Tenga en cuenta que las sentencias GOBACK y EXIT PROGRAM están permitidas en un programa llamado por un procedimiento de entrada o un procedimiento de salida.

Respuesta del programador: Cambie la aplicación para que no utilice uno de los métodos anteriores para finalizar la clasificación o fusión.

Acción del sistema: La aplicación ha terminado.

IWZ013S

Ordenar o fusionar solicitado mientras la clasificación o fusión se ejecuta en una hebra diferente.

Explicación: No se da soporte a la ejecución de la ordenación o fusión en dos o más hebras al mismo tiempo.

Respuesta del programador: ejecute siempre ordenar o fusionar en la misma hebra. De forma alternativa, incluya el código antes de cada llamada a la clasificación o fusión que determina si la clasificación o fusión se está ejecutando en otra hebra. Si la clasificación o fusión se está ejecutando en otra hebra, espere a que la hebra finalice. Si no lo está, establezca un distintivo para indicar que se está ejecutando la clasificación o fusión y llame a la clasificación o fusión.

Acción del sistema: La hebra ha terminado.

IWZ026W

No se ha hecho nunca referencia al registro especial SORT-RETURN, pero el contenido actual ha indicado que la operación de clasificación o fusión en el programa *nombre-programa* en el número de línea *número-línea* no ha sido satisfactoria. El código de retorno de clasificación o fusión era *código de retorno*.

Explicación: El origen COBOL no contiene ninguna referencia al registro SORT-RETURN . El compilador genera una prueba después de cada verbo de clasificación o fusión. La ordenación o fusión ha devuelto al programa un código de retorno distinto de cero.

Respuesta del programador: Determine por qué la ordenación o fusión no ha sido satisfactoria y solucione el problema. Consulte "Ordenar y fusionar números de error" en la página 172 para obtener la lista de posibles códigos de retorno.

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ029S

Argument-1 para la función *nombre-función* en el programa *nombre-programa* en la línea *número-línea* era menor que cero.

Explicación: Se ha utilizado un valor no permitido para argument-1 .

Respuesta del programador: Asegúrese de que el argument-1 sea mayor o igual que cero.

Acción del sistema: La aplicación ha terminado.

IWZ030S

Argument-2 para la función *nombre-función* en el programa *nombre-programa* en la línea *número-línea* no era un entero positivo.

Explicación: Se ha utilizado un valor no permitido para argument-1 .

Respuesta del programador: Asegúrese de que el argument-2 es un entero positivo.

Acción del sistema: La aplicación ha terminado.

IWZ036W

Se ha producido un truncamiento de posiciones de dígitos de orden superior en el programa *nombre-programa* en el número de línea *número-línea*.

Explicación: El código generado ha truncado un resultado intermedio (es decir, el almacenamiento temporal utilizado durante un cálculo aritmético) a 30 dígitos; algunos de los dígitos truncados no eran 0.

Respuesta del programador: Consulte los conceptos relacionados al final de esta sección para obtener una descripción de los resultados intermedios.

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ037I

El flujo de control en el programa *nombre-programa* ha seguido más allá de la última línea del programa. Control devuelto al interlocutor del programa *nombre-programa*.

Explicación: El programa no tenía un terminador (STOP, GOBACK o EXIT) y el control se ha caído a través de la última instrucción.

Respuesta del programador: Compruebe la lógica del programa. A veces este error se produce debido a uno de los siguientes errores lógicos:

- El último párrafo del programa sólo debía recibir el control como resultado de una sentencia PERFORM, pero debido a un error lógico se ramificó en una sentencia GO TO.
- El último párrafo del programa se ejecutó como resultado de una ruta "de paso a través", y no había ninguna sentencia al final del párrafo para finalizar el programa.

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ038S

Se ha encontrado un valor de longitud de modificación de referencia de *valor-modificación-referencia* en la línea *número-línea* que no era igual a 1 en una referencia al elemento de datos *elemento-datos*.

Explicación: El valor de longitud de una especificación de modificación de referencia no era igual a 1. El valor de longitud debe ser igual a 1.

Respuesta del programador: Compruebe el número de línea indicado en el programa para asegurarse de que los valores de longitud modificados de referencia son (o se resolverán en) 1.

Acción del sistema: La aplicación ha terminado.

IWZ039S

Se ha detectado un signo de sobreperforación no válido.

Explicación: El valor de la posición de signo no era válido.

Dado X'*sd*', donde *s* es la representación de signo y *d* representa el dígito, las representaciones de signo válidas para decimal externo (`USAGE DISPLAY` sin la cláusula `SIGN IS SEPARATE`) son:

Positivo:	0, 1, 2, 3, 8, 9, A y B
Negativo:	4, 5, 6, 7, C, D, E y F

Los signos generados internamente son 3 para positivos y sin firmar, y 7 para negativos.

Dado que X'*ds*', donde *d* representa el dígito y *s* es la representación de signo, las representaciones de signo válidas para datos COBOL decimales internos (`USAGE PACKED-DECIMAL`) son:

Positivo:	A, C, E y F

Negativo: B y D

Los signos generados internamente son C para positivos y sin signo, y D para negativos.

Respuesta del programador: Es posible que este error se haya producido debido a una cláusula REDEFINES que implica la posición del signo o un movimiento de grupo que implica la posición del signo, o a que la posición nunca se ha inicializado. Compruebe los casos anteriores.

Acción del sistema: La aplicación ha terminado.

IWZ040S

Se ha detectado un signo separado no válido.

Explicación: Se ha intentado una operación en los datos definidos con un signo aparte. El valor en la posición de signo no era un signo más (+) o un signo menos (-).

Respuesta del programador: Es posible que este error se haya producido debido a una cláusula REDEFINES que implica la posición del signo o un movimiento de grupo que implica la posición del signo, o a que la posición nunca se ha inicializado. Compruebe los casos anteriores.

Acción del sistema: La aplicación ha terminado.

IWZ048W

Se ha elevado una base negativa a una potencia fraccional en una expresión de exponenciación. Se utilizó el valor absoluto de la base.

Explicación: Se ha producido un número negativo elevado a una alimentación fraccional en una rutina de biblioteca.

El valor de un número negativo elevado a una potencia fraccional no está definido en COBOL. Si hubiera aparecido una cláusula SIZE ERROR en la sentencia en cuestión, se habría utilizado el imperativo SIZE ERROR. Sin embargo, no había ninguna cláusula SIZE ERROR, por lo que el valor absoluto de la base se utilizó en la exponenciación.

Respuesta del programador: Asegúrese de que las variables de programa de la sentencia anómala se hayan establecido correctamente.

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ049W

Se ha elevado una base cero a una potencia cero en una expresión de exponenciación. El resultado se ha establecido en uno.

Explicación: El valor de cero elevado a la potencia cero se ha producido en una rutina de biblioteca.

El valor de cero elevado a la potencia cero no está definido en COBOL. Si hubiera aparecido una cláusula SIZE ERROR en la sentencia en cuestión, se habría utilizado el imperativo SIZE ERROR. Sin embargo, no había ninguna cláusula SIZE ERROR presente, por lo que el valor devuelto era uno.

Respuesta del programador: Asegúrese de que las variables de programa de la sentencia anómala se hayan establecido correctamente.

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ050S

Se ha elevado una base cero a una potencia negativa en una expresión de exponenciación.

Explicación: El valor de cero elevado a potencia negativa se ha producido en una rutina de biblioteca.

El valor de cero elevado a un número negativo no está definido. Si hubiera aparecido una cláusula SIZE ERROR en la sentencia en cuestión, se habría utilizado el imperativo SIZE ERROR. Sin embargo, no había ninguna cláusula SIZE ERROR.

Respuesta del programador: Asegúrese de que las variables de programa de la sentencia anómala se hayan establecido correctamente.

Acción del sistema: La aplicación ha terminado.

IWZ051S

No quedan dígitos significativos en una operación de exponenciación de punto fijo en el programa *nombre-programa* debido a que se han especificado demasiadas posiciones decimales en los operandos o receptores.

Explicación: Un cálculo de punto fijo ha producido un resultado que no tenía dígitos significativos porque los operandos o el receptor tenían demasiadas posiciones decimales.

Respuesta del programador: Modifique las cláusulas PICTURE de los operandos o el elemento numérico receptor según sea necesario para tener posiciones enteras adicionales y menos posiciones decimales.

Acción del sistema: La aplicación ha terminado.

IWZ053S

Se ha producido un desbordamiento en la conversión a coma flotante.

Explicación: Se ha generado un número en el programa que es demasiado grande para representarse en coma flotante.

Respuesta del programador: Debe modificar el programa adecuadamente para evitar un desbordamiento.

Acción del sistema: La aplicación ha terminado.

IWZ054S

Se ha producido una excepción de coma flotante.

Explicación: Un cálculo de coma flotante ha generado un resultado no permitido. Los cálculos de coma flotante se realizan utilizando la aritmética de coma flotante IEEE, que puede producir resultados denominados NaN (Not a Number). Por ejemplo, el resultado de 0 dividido por 0 es NaN.

Respuesta del programador: Modifique el programa para probar los argumentos de esta operación para que no se produzca NaN.

Acción del sistema: La aplicación ha terminado.

IWZ055W

Se ha producido un subdesbordamiento en la conversión a coma flotante. El resultado se ha establecido en cero.

Explicación: En la conversión a coma flotante, el exponente negativo ha superado el límite del hardware. El valor de coma flotante se ha establecido en cero.

Respuesta del programador: No es necesaria ninguna acción, aunque es posible que desee modificar el programa para evitar un subdesbordamiento.

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ058S

Se ha producido un desbordamiento de exponente.

Explicación: Se ha producido un desbordamiento de exponente de coma flotante en una rutina de biblioteca.

Respuesta del programador: Asegúrese de que las variables de programa de la sentencia anómala se hayan establecido correctamente.

Acción del sistema: La aplicación ha terminado.

IWZ059W

Se ha truncado un exponente con más de nueve dígitos.

Explicación: los exponentes en exponencias de punto fijo no pueden contener más de nueve dígitos. El exponente se ha truncado de nuevo a nueve dígitos; algunos de los dígitos truncados no eran 0.

Respuesta del programador: No es necesaria ninguna acción, aunque es posible que desee ajustar el exponente en la sentencia anómala.

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ060W

Se ha producido un truncamiento de posiciones de dígitos de orden superior.

Explicación: El código de una rutina de biblioteca ha truncado un resultado intermedio (es decir, el almacenamiento temporal utilizado durante un cálculo aritmético) a 30 dígitos; algunos de los dígitos truncados no eran 0.

Respuesta del programador: Consulte los conceptos relacionados al final de esta sección para obtener una descripción de los resultados intermedios.

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ061S

Se ha producido una división por cero.

Explicación: Se ha producido una división por cero en una rutina de biblioteca. La división por cero no está definida. Si hubiera aparecido una cláusula SIZE ERROR en la sentencia en cuestión, se habría utilizado el imperativo SIZE ERROR. Sin embargo, no había ninguna cláusula SIZE ERROR.

Respuesta del programador: Asegúrese de que las variables de programa de la sentencia anómala se hayan establecido correctamente.

Acción del sistema: La aplicación ha terminado.

IWZ063S

Se ha detectado un signo no válido en un campo de envío editado numérico en *nombre-programa* en el número de línea *número-línea*.

Explicación: Se ha intentado mover un campo editado numérico con signo a un campo receptor numérico con signo o numérico editado en una sentencia MOVE. Sin embargo, la posición del signo en el campo emisor contenía un carácter que no era un carácter de signo válido para la PICTURE correspondiente.

Respuesta del programador: Asegúrese de que las variables de programa de la sentencia anómala se hayan establecido correctamente.

Acción del sistema: La aplicación ha terminado.

IWZ064S

Se ha intentado una llamada recursiva al programa activo *nombre-programa* en la unidad de compilación *unidad-compilación* .

Explicación: COBOL no permite la reinvocación de un programa interno que ha comenzado la ejecución, pero aún no ha terminado. Por ejemplo, si los programas internos A y B son hermanos de un programa contenedor, y A llama a B y B llama a A, se emitirá este mensaje.

Respuesta del programador: Examine el programa para eliminar llamadas a programas internos activos.

Acción del sistema: La aplicación ha terminado.

IWZ065I

Se ha intentado una operación CANCEL del programa activo *nombre-programa* en la unidad de compilación *unidad-compilación* .

Explicación: Se ha intentado cancelar un programa interno activo. Por ejemplo, si los programas internos A y B son hermanos en un programa contenedor y A llama a B y B cancela A, se emitirá este mensaje.

Respuesta del programador: Examine el programa para eliminar la cancelación de programas internos activos.

Acción del sistema: La aplicación ha terminado.

IWZ066S

La longitud del registro de datos externo *registro-datos* en el programa *nombre-programa* no coincidía con la longitud existente del registro.

Explicación: Al procesar registros de datos externos durante la inicialización del programa, se ha determinado que un registro de datos externos se había definido previamente en otro programa de la unidad de ejecución y que la longitud del registro especificada en el programa actual no era la misma que la longitud definida anteriormente.

Respuesta del programador: Examine el archivo actual y asegúrese de que los registros de datos externos se hayan especificado correctamente.

Acción del sistema: La aplicación ha terminado.

IWZ071S

ALL subscripted table reference to table *nombre-tabla* by verb number *número-verbo* on line *número-línea* tenía un subíndice ALL especificado para una dimensión OCCURS DEPENDING ON, y el objeto era menor o igual que 0.

Explicación: Cuando la opción SSRANGE está en vigor, este mensaje se emite para indicar que hay 0 apariciones de dimensión con subscripción de ALL.

La comprobación se realiza en el valor actual del objeto `OCCURS DEPENDING ON`.

Respuesta del programador: Asegúrese de que los objetos ODO de todas las dimensiones con subíndice de cualquier elemento subíndice de la sentencia indicada sean positivos.

Acción del sistema: La aplicación ha terminado.

IWZ072S

Un valor de posición inicial de modificación de referencia de *valor-modificación-referencia* en la línea *número-línea* hacía referencia a un área fuera de la región del elemento de datos *elemento-datos*.

Explicación: El valor de la posición inicial en una especificación de modificación de referencia era menor que 1, o era mayor que la longitud actual del elemento de datos que se estaba modificando de referencia. El valor de posición inicial debe ser un entero positivo menor o igual que el número de caracteres del elemento de datos modificado de referencia.

Respuesta del programador: Compruebe el valor de la posición inicial en la especificación de modificación de referencia.

Acción del sistema: La aplicación ha terminado.

IWZ073S

Se ha encontrado un valor de longitud de modificación de referencia no positiva de *valor-modificación-referencia* en la línea *número-línea* en una referencia al elemento de datos *elemento-datos*.

Explicación: El valor de longitud de una especificación de modificación de referencia era menor o igual que 0. El valor de longitud debe ser un entero positivo.

Respuesta del programador: Compruebe el número de línea indicado en el programa para asegurarse de que los valores de longitud modificados de referencia son (o se resolverán) enteros positivos.

Acción del sistema: La aplicación ha terminado.

IWZ074S

Un valor de posición inicial de modificación de referencia de *valor-modificación-referencia* y un valor de longitud de *longitud* en línea *número-línea* ha hecho que la referencia se realice más allá del carácter situado más a la derecha del elemento de datos *elemento de datos*.

Explicación: La posición inicial y el valor de longitud de una especificación de modificación de referencia se combinan para direccionar un área más allá del final del elemento de datos modificado de referencia. La suma de la posición inicial y el valor de longitud menos uno debe ser menor o igual que el número de caracteres en el elemento de datos modificado de referencia.

Respuesta del programador: Compruebe el número de línea indicado en el programa para asegurarse de que los valores de inicio y longitud modificados de referencia estén establecidos de forma que no se haga una referencia más allá del carácter situado más a la derecha del elemento de datos.

Acción del sistema: La aplicación ha terminado.

IWZ075S

Se han encontrado incoherencias en el archivo EXTERNAL *nombre-archivo* en el programa *nombre-programa*. Los atributos de archivo siguientes no coinciden con los del archivo externo establecido: *attribute-1 attribute-2 attribute-3 attribute-4 attribute-5 attribute-6 attribute-7*.

Explicación: Uno o varios atributos de un archivo externo no coinciden entre dos programas que lo han definido.

Respuesta del programador: Corrija el archivo externo. Para obtener un resumen de los atributos de archivo que deben coincidir entre las definiciones del mismo archivo externo, consulte la publicación *COBOL for Linux en x86 Consulta de lenguaje*.

Acción del sistema: La aplicación ha terminado.

IWZ076W

El número de caracteres en el nombre de datos INSPECT SUSTITUT CHARACTERS BY no era igual a uno. Se ha utilizado el primer carácter.

Explicación: Un elemento de datos que aparece en una frase CHARACTERS dentro de una frase REPLACING en una sentencia INSPECT debe definirse como un carácter de longitud. Debido a una especificación de modificación de referencia para este elemento de datos, el valor de longitud resultante no era igual a uno. Se supone que el valor de longitud es uno.

Respuesta del programador: Puede corregir las especificaciones de modificación de referencia en la sentencia INSPECT que falla para asegurarse de que la longitud de modificación de referencia es (o se resolverá en) 1; la acción del programador no es necesaria.

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ077W

Las longitudes de los elementos de datos INSPECT no eran iguales. Se ha utilizado la longitud más corta.

Explicación: Los dos elementos de datos que aparecen en una frase REPLACING o CONVERTING en una sentencia INSPECT deben tener longitudes iguales, excepto cuando el segundo elemento de este tipo es una constante figurativa. Debido a la modificación de referencia para uno o ambos de estos elementos de datos, los valores de longitud resultantes no eran iguales. El valor de longitud más corta se aplica a ambos elementos y la ejecución continúa.

Respuesta del programador: Puede ajustar los operandos de longitud desigual en la sentencia INSPECT anómala; no es necesaria la acción del programador.

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ078S

ALL subscripted table reference to table *nombre-tabla* por número de verbo *número-verbo* en línea *número-línea* superará el límite superior de la tabla.

Explicación: Cuando la opción SSRANGE está en vigor, este mensaje se emite para indicar que una tabla multidimensional con ALL especificado como uno o más de los subíndices dará como resultado una referencia más allá del límite superior de la tabla.

La comprobación de rango se ha realizado en el compuesto de los subíndices y el número máximo de apariciones para las dimensiones con subíndice ALL. Para las tablas de longitud variable, la dirección está fuera de la región de la tabla definida cuando todos los objetos OCCURS DEPENDING ON están en sus valores máximos; no se tiene en cuenta el valor actual del objeto ODO. La comprobación no se ha realizado en subíndices individuales.

Respuesta del programador: Asegúrese de que los objetos OCCURS DEPENDING ON evaluados en tiempo de ejecución no excedan el número máximo de apariciones de la dimensión para los elementos de tabla a los que se hace referencia en la sentencia anómala.

Acción del sistema: La aplicación ha terminado.

IWZ096C

Las variantes de mensaje incluyen:

- **La llamada dinámica del programa *nombre-programa* ha fallado. Una carga de módulo *nombre-módulo* ha fallado con un código de error de *cód-error*.**
- **La llamada dinámica del programa *nombre-programa* ha fallado. Una carga del módulo *nombre-módulo* ha fallado con un código de retorno de *cód-retorno*.**
- **La llamada dinámica del programa *nombre-programa* ha fallado. Recursos insuficientes.**
- **La llamada dinámica del programa *nombre-programa* ha fallado. COBPATH no se ha encontrado en el entorno.**
- **La llamada dinámica del programa *nombre-programa* ha fallado. No se ha encontrado la entrada *nombre-entrada* .**
- **La llamada dinámica ha fallado. El nombre del programa de destino no contiene ningún carácter válido.**
- **La llamada dinámica del programa *nombre-programa* ha fallado. El módulo de carga *módulo de carga* no se ha podido encontrar en los directorios identificados en la variable de entorno COBPATH.**

Explicación: Una llamada dinámica ha fallado debido a una de las razones listadas en las variantes de mensaje anteriores. En lo anterior, el valor de *código de error* es el número de error establecido por `load`.

Respuesta del programador: Compruebe que COBPATH esté definido. Compruebe que el módulo existe. Compruebe que el nombre del módulo que se va a cargar coincide con el nombre de la entrada llamada. Compruebe que el módulo que se va a cargar se ha creado correctamente utilizando las `cob2` opciones.

Acción del sistema: La aplicación ha terminado.

IWZ097S

Argument-1 para la función *nombre-función* no contenía dígitos.

Explicación: Argument-1 para la función indicada debe contener al menos 1 dígito.

Respuesta del programador: Ajuste el número de dígitos en Argument-1 en la sentencia anómala.

Acción del sistema: La aplicación ha terminado.

IWZ100S

Argument-1 para la función *función* era menor o igual que -1.

Explicación: Se ha utilizado un valor no permitido para Argument-1.

Respuesta del programador: Asegúrese de que el argument-1 sea mayor que -1.

Acción del sistema: La aplicación ha terminado.

IWZ103S

Argument-1 para la función *nombre-función* era menor que cero o mayor que 99.

Explicación: Se ha utilizado un valor no permitido para Argument-1.

Respuesta del programador: Compruebe que el argumento de la función está en el rango válido.

Acción del sistema: La aplicación ha terminado.

IWZ104S

Argument-1 para la función *nombre-función* era menor que cero o mayor que 99999.

Explicación: Se ha utilizado un valor no permitido para Argument-1.

Respuesta del programador: Compruebe que el argumento de la función está en el rango válido.

Acción del sistema: La aplicación ha terminado.

IWZ105S

Argument-1 para la función *nombre-función* era menor que cero o mayor que 999999.

Explicación: Se ha utilizado un valor no permitido para Argument-1.

Respuesta del programador: Compruebe que el argumento de la función está en el rango válido.

Acción del sistema: La aplicación ha terminado.

IWZ151S

Argument-1 para la función *nombre-función* contenía más de 18 dígitos.

Explicación: El número total de dígitos en Argument-1 de la función indicada ha superado los 18 dígitos.

Respuesta del programador: Ajuste el número de dígitos en Argument-1 en la sentencia anómala.

Acción del sistema: La aplicación ha terminado.

IWZ152S

Carácter no válido *carácter* se ha encontrado en la columna *número-columna* en el argument-1 para la función *nombre-función*.

Explicación: Se ha encontrado un carácter no dígito que no es un separador decimal, coma, espacio o signo (+,-, CR, DB) en el argument-1 para la función NUMVAL/NUMVAL-C.

Respuesta del programador: Corrija el argument-1 para NUMVAL o NUMVAL-C en la sentencia indicada.

Acción del sistema: La aplicación ha terminado.

IWZ155S

Carácter no válido *carácter* se ha encontrado en la columna *número-columna* en el argument-2 para la función *nombre-función*.

Explicación: Se ha encontrado un carácter no permitido en argument-2 para la función NUMVAL-C.

Respuesta del programador: Compruebe que el argumento de función sigue las reglas de sintaxis.

Acción del sistema: La aplicación ha terminado.

IWZ156S

Argument-1 para la función *nombre-función* era menor que cero o mayor que 28.

Explicación: El argumento de entrada para la función FACTORIAL es mayor que 28 o menor que 0.

Respuesta del programador: Compruebe que el argumento de la función está en el rango válido.

Acción del sistema: La aplicación ha terminado.

IWZ157S

La longitud de Argument-1 para la función *nombre-función* no era igual a 1.

Explicación: La longitud del argumento de entrada para la función ORD no es 1.

Respuesta del programador: Compruebe que el argumento de función tenga sólo 1 byte de longitud.

Acción del sistema: La aplicación ha terminado.

IWZ158S

Argument-1 para la función *nombre-función* era menor que cero o mayor que 29.

Explicación: El argumento de entrada para la función FACTORIAL es mayor que 29 o menor que 0.

Respuesta del programador: Compruebe que el argumento de la función está en el rango válido.

Acción del sistema: La aplicación ha terminado.

IWZ159S

Argument-1 para la función *nombre-función* era menor que 1 o mayor que 3067671.

Explicación: El argumento de entrada para la función DATE-OF-INTEGER o DAY-OF-INTEGER es menor que 1 o mayor que 3067671.

Respuesta del programador: Compruebe que el argumento de la función está en el rango válido.

Acción del sistema: La aplicación ha terminado.

IWZ160S

Argument-1 para la función *nombre-función* era menor que 16010101 o mayor que 9999991231.

Explicación: El argumento de entrada para la función INTEGER-OF-DATE es menor que 16010101 o mayor que 9999991231.

Respuesta del programador: Compruebe que el argumento de la función está en el rango válido.

Acción del sistema: La aplicación ha terminado.

IWZ161S

Argument-1 para la función *nombre-función* era menor que 1601001 o mayor que 9999365.

Explicación: El argumento de entrada para la función INTEGER-OF-DAY es menor que 1601001 o mayor que 999999365.

Respuesta del programador: Compruebe que el argumento de la función está en el rango válido.

Acción del sistema: La aplicación ha terminado.

IWZ162S

Argument-1 para la función *nombre-función* era menor que 1 o mayor que el número de posiciones en el orden de clasificación del programa.

Explicación: El argumento de entrada para la función CHAR es menor que 1 o mayor que la posición ordinal más alta en la secuencia de clasificación del programa.

Respuesta del programador: Compruebe que el argumento de la función está en el rango válido.

Acción del sistema: La aplicación ha terminado.

IWZ163S

Argument-1 para la función *nombre-función* era menor que cero.

Explicación: El argumento de entrada para la función RANDOM es menor que 0.

Respuesta del programador: Corrija el argumento de la función RANDOM en la sentencia anómala.

Acción del sistema: La aplicación ha terminado.

IWZ165S

Un valor de posición inicial de modificación de referencia de *valor-posición-inicio* en la línea *número de línea* hacía referencia a un área fuera de la región del resultado de la función de *resultado-función*.

Explicación: El valor de la posición inicial en una especificación de modificación de referencia era menor que 1, o era mayor que la longitud actual del resultado de función que se estaba modificando de referencia. El valor de posición inicial debe ser un entero positivo menor o igual que el número de caracteres en el resultado de la función modificada de referencia.

Respuesta del programador: Compruebe el valor de la posición inicial en la especificación de modificación de referencia y la longitud del resultado de la función real.

Acción del sistema: La aplicación ha terminado.

IWZ166S

Se ha encontrado un valor de longitud de modificación de referencia no positiva de *longitud* en la línea *número-línea* en una referencia al resultado de la función de *resultado-función*.

Explicación: El valor de longitud de una especificación de modificación de referencia para un resultado de función era menor o igual que 0. El valor de longitud debe ser un entero positivo.

Respuesta del programador: Compruebe el valor de longitud y realice la corrección adecuada.

Acción del sistema: La aplicación ha terminado.

IWZ167S

Un valor de posición inicial de modificación de referencia de *posición inicial* y un valor de longitud de *longitud* en línea *número-línea* han hecho que la referencia se realice más allá del carácter situado más a la derecha del resultado de la función de *resultado-función*.

Explicación: La posición inicial y el valor de longitud de una especificación de modificación de referencia se combinan para direccionar un área más allá del final del resultado de la función modificada de referencia. La suma de la posición inicial y el valor de longitud menos uno debe ser menor o igual que el número de caracteres en el resultado de la función modificada de referencia.

Respuesta del programador: Compruebe la longitud de la especificación de modificación de referencia con respecto a la longitud real del resultado de la función y realice las correcciones adecuadas.

Acción del sistema: La aplicación ha terminado.

IWZ168W

SYSPUNCH/SYSPCH tomará por omisión el dispositivo de salida lógica del sistema. No se ha establecido la variable de entorno correspondiente.

Explicación: Los nombres de entorno COBOL (como SYSPUNCH/SYSPCH) se utilizan como los nombres de variable de entorno correspondientes a los nombres mnemónicos utilizados en las sentencias ACCEPT y DISPLAY. Establézcalos igual a archivos, no a nombres de directorio existentes. Para establecer variables de entorno, utilice el mandato `export`.

Puede establecer variables de entorno de forma persistente o temporal.

Respuesta del programador: Si no desea que SYSPUNCH/SYSPCH tenga el valor predeterminado en la pantalla, establezca la variable de entorno correspondiente.

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ169S

Tipo de dispositivo desconocido para la sentencia DISPLAY.

Explicación: Se ha especificado un tipo de dispositivo desconocido en *environment-name-1* o el nombre de entorno asociado con *mnemonic-name-1* de la sentencia DISPLAY.

Respuesta del programador: especifique un tipo de dispositivo válido. Para ver los tipos válidos, consulte el párrafo SPECIAL-NAMES.

Acción del sistema: La aplicación ha terminado.

IWZ170S

Tipo de datos no permitido para el operando DISPLAY.

Explicación: Se ha especificado un tipo de datos no válido como destino de la sentencia DISPLAY.

Respuesta del programador: especifique un tipo de datos válido. Los siguientes tipos de datos **no** son válidos:

- Elementos de datos definidos con `USAGE IS FUNCTION-POINTER`
- Elementos de datos definidos con `USAGE IS PROCEDURE-POINTER`
- Elementos de datos o nombres de índice definidos con `USAGE IS INDEX`

Acción del sistema: La aplicación ha terminado.

IWZ171I

nombre-serie no es una opción de tiempo de ejecución válida.

Explicación: *nombre-serie* no es una opción válida.

Respuesta del programador: `CHECK`, `DEBUG`, `ERRCOUNT`, `FILESYS`, `TRAP`y `UPSI` son opciones de tiempo de ejecución válidas.

Acción del sistema: *nombre-serie* se ignora.

IWZ172I

La serie _nombre-serie_ no es una subopción válida de la opción de tiempo de ejecución _nombre-opción_.

Explicación: *nombre-serie* no estaba en el conjunto de valores reconocidos.

Respuesta del programador: Elimine la subopción no válida *serie* de la opción de tiempo de ejecución *nombre-opción*.

Acción del sistema: La subopción no válida se ignora.

IWZ173I

La serie de subopción _nombre-serie_ de la opción de tiempo de ejecución _nombre-opción_ debe tener _número de_ caracteres de longitud. Se utilizará el valor predeterminado.

Explicación: El número de caracteres para la serie de subopción *nombre-serie* de la opción de tiempo de ejecución *nombre-opción* no es válido.

Respuesta del programador: Si no desea aceptar el valor predeterminado, especifique una longitud de caracteres válida.

Acción del sistema: se utilizará el valor predeterminado.

IWZ174I

La serie de subopción _nombre-serie_ de la opción de tiempo de ejecución _nombre-opción_ contiene uno o más caracteres no válidos. Se utilizará el valor predeterminado.

Explicación: Se ha detectado al menos un carácter no válido en la subopción especificada.

Respuesta del programador: Si no desea aceptar el valor predeterminado, especifique caracteres válidos.

Acción del sistema: se utilizará el valor predeterminado.

IWZ175S

No hay soporte para la rutina *nombre-rutina* en este sistema.

Explicación: *nombre-rutina* no está soportado.

Respuesta del programador:

Acción del sistema: La aplicación ha terminado.

IWZ176S

Argument-1 para la función *nombre-función* era mayor que *valor-decimal*.

Explicación: Se ha utilizado un valor no permitido para argument-1 .

Respuesta del programador: Asegúrese de que argument-1 sea menor o igual que *valor-decimal*.

Acción del sistema: La aplicación ha terminado.

IWZ177S

Argument-2 para la función *nombre-función* era igual a *valor-decimal*.

Explicación: Se ha utilizado un valor no permitido para argument-2 .

Respuesta del programador: Asegúrese de que el argument-1 no es igual a *valor-decimal*.

Acción del sistema: La aplicación ha terminado.

IWZ178S

Argument-1 para la función *nombre-función* era menor o igual que *valor-decimal*.

Explicación: Se ha utilizado un valor no permitido para Argument-1 .

Respuesta del programador: Asegúrese de que Argument-1 sea mayor que *valor-decimal*.

Acción del sistema: La aplicación ha terminado.

IWZ179S

Argument-1 para la función *nombre-función* era menor que *valor-decimal*.

Explicación: Se ha utilizado un valor no permitido para Argument-1 .

Respuesta del programador: Asegúrese de que Argument-1 sea igual o mayor que *valor-decimal*.

Acción del sistema: La aplicación ha terminado.

IWZ180S

Argument-1 para la función *nombre-función* no era un entero.

Explicación: Se ha utilizado un valor no permitido para Argument-1 .

Respuesta del programador: Asegúrese de que Argument-1 sea un entero.

Acción del sistema: La aplicación ha terminado.

IWZ181I

Se ha encontrado un carácter no válido en la serie numérica *serie* de la opción de tiempo de ejecución *nombre-opción*. Se utilizará el valor predeterminado.

Explicación: *serie* no contenía todos los caracteres numéricos decimales.

Respuesta del programador: Si no desea el valor predeterminado, corrija la serie de la opción de tiempo de ejecución para que contenga todos los caracteres numéricos.

Acción del sistema: Se utilizará el valor predeterminado.

IWZ182I

El número *número* de la opción de tiempo de ejecución *nombre-opción* ha superado el rango de *min-range* a *max-range*. Se utilizará el valor predeterminado.

Explicación: *número* ha superado el rango de *min-range* a *max-range*.

Respuesta del programador: Corrija la serie de la opción de tiempo de ejecución para que esté dentro del rango válido.

Acción del sistema: Se utilizará el valor predeterminado.

IWZ183S

El nombre de función en _iwzCOBOLInit ha devuelto un resultado.

Explicación: La rutina de salida de terminación de unidad de ejecución ha devuelto a la función que ha invocado la rutina (la función especificada en `function_code`).

Respuesta del programador: Vuelva a escribir la función para que la rutina de salida de terminación de unidad de ejecución efectúe un salto de longitud o una salida en lugar de volver a la función.

Acción del sistema: La aplicación ha terminado.

IWZ200S

Se ha detectado un error durante la *operación de E/S* para el archivo *nombre-archivo*. El estado del archivo es: *file-status*.

Explicación: Se ha detectado un error durante una operación de E/S de archivo. No se ha especificado ningún estado de archivo para el archivo y no hay ninguna declaración de error aplicable en vigor para el archivo.

Respuesta del programador: Corrija la condición descrita en este mensaje. Puede especificar la cláusula `FILE STATUS` para el archivo si desea detectar el error y realizar las acciones adecuadas dentro del programa fuente.

Acción del sistema: La aplicación ha terminado.

IWZ200S

STOP o ACCEPT han fallado con un error de E/S, *código de error*. La unidad de ejecución se cancela.

Explicación: Ha fallado una sentencia STOP o ACCEPT.

Respuesta del programador: Compruebe que STOP o ACCEPT hace referencia a un archivo o dispositivo de terminal legítimo.

Acción del sistema: La aplicación ha terminado.

IWZ201C

```
Message variants include:
Access Intent List Error.
Concurrent Opens Exceeds Maximum.
Cursor Not Selecting a Record Position.
Data Stream Syntax Error.
Duplicate Key Different Index.
Duplicate Key Same Index.
Duplicate Record Number.
File Temporarily Not Available.
File system cannot be found.
File Space Not Available.
File Closed with Damage.
Invalid Key Definition.
Invalid Base File Name.
Key Update Not Allowed by Different Index.
Key Update Not Allowed by Same Index.
No Update Intent on Record.
Not Authorized to Use Access Method.
Not Authorized to Directory.
Not Authorized to Function.
Not authorized to File.
Parameter Value Not Supported.
Parameter Not Supported.
Record Number Out of Bounds.
Record Length Mismatch.
Resource Limits Reached in Target System.
Resource Limits Reached in Source System.
```

```
Address Error.
Command Check.
Duplicate File Name.
End of File Condition.
Existing Condition.
File Handle Not Found.
Field Length Error.
File Not Found.
File Damaged.
File is Full.
File In Use.
Function Not Supported.
Invalid Access Method.
Invalid Data Record.
Invalid Key Length.
Invalid File Name.
Invalid Request.
Invalid Flag.
Object Not Supported.
Record Not Available.
Record Not Found.
Record Inactive.
Record Damaged.
Record In Use.
Update Cursor Error.
```

Explicación: Se ha detectado un error durante una operación de E/S para un archivo STL . No se ha especificado ningún estado de archivo para el archivo y no hay ninguna declaración de error aplicable en vigor para el archivo.

Respuesta del programador: Corrija la condición descrita en este mensaje.

Acción del sistema: La aplicación ha terminado.

IWZ203S

La página de códigos en vigor no es una página de códigos DBCS.

Explicación: Las referencias a datos DBCS se han realizado con una página de códigos no DBCS en vigor.

Respuesta del programador: Para datos DBCS, especifique una página de códigos DBCS válida. Las páginas de códigos DBCS válidas son:

País o región	**Página de códigos**
Japón	IBM-932
Corea	IBM-1363
República Popular China (simplificada)	IBM-1386
Taiwán (tradicional)	

Nota: es posible que las páginas de códigos listadas anteriormente no estén soportadas para una versión o release específico de dicha plataforma.

Acción del sistema: La aplicación ha terminado.

IWZ204S

Se ha producido un error durante la conversión de DBCS ASCII a DBCS EBCDIC.

Explicación: Una prueba de clase Kanji o DBCS ha fallado debido a un error detectado durante la conversión de serie EBCDIC de serie de caracteres ASCII.

Respuesta del programador: Verifique que el entorno local en vigor sea coherente con la serie de caracteres ASCII que se está probando. Es probable que no sea necesaria ninguna acción si el valor de entorno local es correcto. Es probable que la prueba de clase indique que la serie no es Kanji o no DBCS correctamente.

Acción del sistema: La aplicación ha terminado.

IWZ221S

No se puede abrir el conversor de ICU para la página de códigos, *valor de página de códigos*. El código de error es *valor de código de error*.

Explicación: No se puede abrir el conversor ICU para convertir entre la página de códigos y UTF-16 .

Respuesta del programador: Verifique que el valor de la página de códigos identifica un nombre de página de códigos primario o alias soportado por las bibliotecas de conversión ICU (consulte *Componentes internacionales para Unicode: Explorador de conversores*). Si el valor de la página de códigos es válido, póngase en contacto con el representante de IBM .

Acción del sistema: La aplicación ha terminado.

IWZ222S

La conversión de datos a través de ICU ha fallado con el código de error *valor de código de error*.

Explicación: La conversión de datos a través de ICU ha fallado.

Respuesta del programador: Póngase en contacto con el representante de IBM .

Acción del sistema: La aplicación ha terminado.

IWZ223W

El cierre del conversor ICU ha fallado con el código de error *valor de código de error*.

Explicación: El cierre de un convertidor ICU ha fallado.

Respuesta del programador: Póngase en contacto con el representante de IBM .

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ224S

No se puede abrir el collator de ICU para el valor de entorno local, *valor de entorno local*. El código de error es *valor de código de error*.

Explicación: No se puede abrir el collator de ICU para el entorno local.

Respuesta del programador: Póngase en contacto con el representante de IBM .

Acción del sistema: La aplicación ha terminado.

IWZ225S

Función de correlación de casos Unicode que utiliza ICU ha fallado con el código de error *valor de código de error*. El entorno local en vigor es *valor de entorno local*.

Explicación: La función de correlación de casos de ICU ha fallado.

Respuesta del programador: Póngase en contacto con el representante de IBM .

Acción del sistema: La aplicación ha terminado.

IWZ230W

La tabla de conversión para la páginade códigos actual, *ASCII codeset-id*, a la páginade códigos EBCDIC, *EBCDIC codeset-id*, no está disponible. Se utilizará la tabla de conversión ASCII a EBCDIC predeterminada.

Explicación: La aplicación tiene un módulo compilado con la opción de compilador `CHAR(EBCDIC)` . En tiempo de ejecución, se construirá una tabla de conversión para manejar la conversión de la página de códigos ASCII actual a una página de códigos EBCDIC especificada por la variable de entorno EBCDIC_CODEPAGE. Este error se ha producido porque una tabla de conversión no está disponible para las páginas de códigos especificadas o la especificación de la página EBCDIC_CODE no es válida. La ejecución continuará con una tabla de conversión predeterminada basada en la página de códigos ASCII IBM-1252 o equivalente y la página de códigos EBCDIC IBM-037 o equivalente.

Respuesta del programador: Verifique que la variable de entorno EBCDIC_CODEPAGE tenga un valor válido.

Si no se establece EBCDIC_CODEPAGE, se utilizará el valor predeterminado, IBM-037. Esta es la página de códigos predeterminada que utiliza Enterprise COBOL for z/OS.

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ230W

La página de códigos EBCDIC especificada, *Página de códigos EBCDIC*, no es coherente con el entorno local *entorno local*, pero se utilizará según se solicite.

Explicación: La aplicación tiene un módulo compilado con la opción de compilador `CHAR(EBCDIC)` . Este error se ha producido porque la página de códigos especificada no es el mismo idioma que el entorno local actual.

Respuesta del programador: Verifique que la variable de entorno EBCDIC_CODEPAGE sea válida para este entorno local.

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ230W

La página de códigos EBCDIC especificada, *página de códigos EBCDIC*, no está soportada. Se utilizará la página de códigos EBCDIC predeterminada, *Página de códigos EBCDIC*.

Explicación: La aplicación tiene un módulo compilado con la opción de compilador `CHAR(EBCDIC)` . Este error se ha producido porque la especificación de la variable de entorno EBCDIC_CODEPAGE no es válida. La ejecución continuará con la página de códigos de host predeterminada que corresponde al entorno local actual.

Respuesta del programador: Verifique que la variable de entorno EBCDIC_CODEPAGE tenga un valor válido.

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ230S

No se puede abrir la tabla de conversión EBCDIC.

Explicación: La instalación del sistema actual no incluye la tabla de conversión para las páginas de códigos ASCII y EBCDIC predeterminadas.

Respuesta del programador: Vuelva a instalar el compilador y el tiempo de ejecución. Si el problema persiste, póngase en contacto con el representante de IBM .

Acción del sistema: La aplicación ha terminado.

IWZ230S

No se puede crear la tabla de conversión EBCDIC.

Explicación: Se ha abierto la tabla de conversión de ASCII a EBCDIC, pero la conversión ha fallado.

Respuesta del programador: Vuelva a intentar la ejecución desde una ventana nueva.

Acción del sistema: La aplicación ha terminado.

IWZ230S

El programa principal se ha compilado con el distintivo `-host` y la opción `CHAR(NATIVE)` , que no son compatibles.

Explicación: Compilación con el distintivo `-host` y la opción `CHAR(NATIVE)` no está soportada.

Respuesta del programador: Elimine el distintivo `-host` o elimine la opción `CHAR(NATIVE)` . El distintivo `-host` establece `CHAR(EBCDIC)`.

Acción del sistema: La aplicación ha terminado.

IWZ231S

La consulta del valor de entorno local actual ha fallado.

Explicación: Una consulta del entorno de ejecución no ha podido identificar un valor de entorno local válido. Es necesario establecer el entorno local actual para acceder a los archivos de mensajes adecuados y establecer el orden de clasificación. También lo utilizan los servicios de fecha/hora y para el soporte de caracteres EBCDIC.

Respuesta del programador: Compruebe los valores de las siguientes variables de entorno variable:

LANG
Debe establecerse en un entorno local que se haya instalado en la máquina. Especifique `locale -a` para obtener una lista de los valores válidos. El valor predeterminado es en_US.

Acción del sistema: La aplicación ha terminado.

IWZ232W

Las variantes de mensaje incluyen:

- **Se ha producido un error durante la conversión del elemento de datos *nombre-datos* a EBCDIC en el programa *nombre-programa* en el número de línea *valor-decimal*.**
- **Se ha producido un error durante la conversión del elemento de datos *nombre-datos* a ASCII en el programa *nombre-programa* en el número de línea *valor-decimal*.**
- **Se ha producido un error durante la conversión a EBCDIC para el elemento de datos *nombre-datos* en el programa *nombre-programa* en el número de línea *valor-decimal*.**
- **Se ha producido un error durante la conversión a ASCII para el elemento de datos *nombre-datos* en el programa *nombre-programa* en el número de línea *valor-decimal*.**
- **Se ha producido un error durante la conversión de ASCII a EBCDIC en el programa *nombre-programa* en el número de línea *valor-decimal*.**
- **Se ha producido un error durante la conversión de EBCDIC a ASCII en el programa *nombre-programa* en el número de línea *valor-decimal*.**

Explicación: Los datos de un identificador no se han podido convertir entre los formatos ASCII y EBCDIC tal como solicita la opción de compilador `CHAR(EBCDIC)` .

Respuesta del programador: Compruebe que se hayan instalado y seleccionado los entornos locales ASCII y EBCDIC adecuados. Compruebe que los datos del identificador son válidos y se pueden representar en formato ASCII y EBCDIC.

Acción del sistema: No se ha realizado ninguna acción del sistema. Los datos permanecen en su formato no convertido.

IWZ240S

El año base para el programa *nombre-programa* estaba fuera del rango válido de 1900 a 1999. El valor de ventana deslizante *valor-ventana* ha dado como resultado un año base de *año-base*.

Explicación: Cuando se calculó la ventana de 100 años utilizando el año actual y el valor de ventana deslizante especificado con la opción de compilador YEARWINDOW, el año base de la ventana de 100 años estaba fuera del rango válido de 1900 a 1999.

Respuesta del programador: Examine el diseño de la aplicación para determinar si dará soporte a un cambio en el valor de la opción YEARWINDOW. Si la aplicación puede ejecutarse con un cambio en el valor de la opción YEARWINDOW, compile el programa con un valor de opción YEARWINDOW adecuado. Si la aplicación no puede ejecutarse con un cambio en el valor de la opción YEARWINDOW, convierta todos los campos de fecha en fechas expandidas y compile el programa con NODATEPROC.

Acción del sistema: La aplicación ha terminado.

IWZ241S

El año actual estaba fuera de la ventana de 100 años, de *inicio de año* a *fin de año*, para el programa *nombre-programa*.

Explicación: El año actual estaba fuera de la ventana fija de 100 años especificada por el valor de la opción de compilador YEARWINDOW.

Por ejemplo, si un programa COBOL se compila con YEARWINDOW (1920), la ventana de 100 años para el programa es de 1920 a 2019. Cuando el programa se ejecuta en el año 2020, este mensaje de error se produciría ya que el año actual no está dentro de la ventana de los 100 años.

Respuesta del programador: Examine el diseño de la aplicación para determinar si dará soporte a un cambio en el valor de la opción YEARWINDOW. Si la aplicación puede ejecutarse con un cambio en el valor de la opción YEARWINDOW, compile el programa con un valor de opción YEARWINDOW adecuado.

Si la aplicación no puede ejecutarse con un cambio en el valor de la opción YEARWINDOW, convierta todos los campos de fecha en fechas expandidas y compile el programa con NODATEPROC.

Acción del sistema: La aplicación ha terminado.

IWZ242S

Se ha producido un intento no válido de iniciar una sentencia XML PARSE.

Explicación: Una sentencia XML PARSE iniciada por un programa COBOL ya estaba en curso cuando el mismo programa COBOL intentó otra sentencia XML PARSE. Sólo puede estar activa una sentencia XML PARSE en una invocación determinada de un programa COBOL.

Respuesta del programador: Cambie la aplicación para que no inicie otra sentencia XML PARSE desde el mismo programa COBOL.

Acción del sistema: La aplicación ha terminado.

IWZ243S

Se ha producido un intento no válido de finalizar una sentencia XML PARSE.

Explicación: Una sentencia XML PARSE iniciada por un programa COBOL estaba en curso y se ha intentado una de las acciones siguientes:

- Se ha emitido una sentencia GOBACK o EXIT PROGRAM dentro del programa COBOL que ha iniciado la sentencia XML PARSE.
- Un manejador de usuario asociado con el programa que ha iniciado la sentencia XML PARSE ha movido el cursor de reanudación del manejador de condiciones y ha reanudado la aplicación.

Respuesta del programador: Cambie la aplicación para que no utilice uno de los métodos anteriores para finalizar la sentencia XML PARSE.

Acción del sistema: La aplicación ha terminado.

IWZ813S

No había suficiente almacenamiento disponible para satisfacer una solicitud de obtención de almacenamiento.

Explicación: No había suficiente almacenamiento libre disponible para satisfacer una solicitud de obtención de almacenamiento o reasignación. Este mensaje indica que la gestión de almacenamiento no ha podido obtener suficiente almacenamiento del sistema operativo.

Respuesta del programador: Asegúrese de que tiene suficiente almacenamiento disponible para ejecutar la aplicación.

Acción del sistema: No se asigna almacenamiento.

Código de comentarios simbólicos: CEE0PD

IWZ901S

Las variantes de mensaje incluyen:

- **Salidas de programa debidas a un error grave o crítico.**
- **Salidas de programa: se han producido más de ERRCOUNT errores.**

Explicación: Cada mensaje grave o crítico va seguido de un mensaje IWZ901 . También se emite un mensaje IWZ901 si ha utilizado la opción de tiempo de ejecución ERRCOUNT y el número de mensajes de aviso excede ERRCOUNT.

Respuesta del programador: Consulte el mensaje grave o crítico, o aumente ERRCOUNT.

Acción del sistema: La aplicación ha terminado.

IWZ902S

El sistema ha detectado una excepción Decimal-divide .

Explicación: Se ha detectado un intento de dividir un número por 0.

Respuesta del programador: Modifique el programa. Por ejemplo, añada ON SIZE ERROR a la sentencia marcada.

Acción del sistema: La aplicación ha terminado.

IWZ903S

El sistema ha detectado una excepción de datos.

Explicación: Una operación en datos decimales empaquetados o decimales con zona ha fallado porque los datos contenían un valor no válido.

Respuesta del programador: Verifique que los datos sean datos decimales empaquetados o decimales con zona válidos.

Acción del sistema: La aplicación ha terminado.

IWZ907S

Las variantes de mensaje incluyen:

- **Almacenamiento insuficiente.**
- **Almacenamiento insuficiente. No se pueden obtener *number-bytes* bytes de espacio para el *almacenamiento*.**

Explicación: La biblioteca de tiempo de ejecución ha solicitado espacio de memoria virtual y el sistema operativo ha denegado la solicitud.

Respuesta del programador: El programa utiliza una gran cantidad de memoria virtual y se ha quedado sin espacio. El problema normalmente no se debe a una sentencia en particular, sino que está asociado con el programa en su conjunto. Examine el uso de las cláusulas OCCURS y reduzca el tamaño de las tablas.

Acción del sistema: La aplicación ha terminado.

IWZ993W

Almacenamiento insuficiente. No se puede encontrar espacio para el mensaje *número-mensaje*.

Explicación: La biblioteca de tiempo de ejecución ha solicitado espacio de memoria virtual y el sistema operativo ha denegado la solicitud.

Respuesta del programador: El programa utiliza una gran cantidad de memoria virtual y se ha quedado sin espacio. El problema normalmente no se debe a una sentencia en particular, sino que está asociado

con el programa en su conjunto. Examine el uso de las cláusulas OCCURS y reduzca el tamaño de las tablas.

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ994W

No se puede encontrar el mensaje *message-number* en *message-catalog*.

Explicación: La biblioteca de tiempo de ejecución no puede encontrar el catálogo de mensajes o un mensaje determinado en el catálogo de mensajes.

Respuesta del programador: Compruebe que la biblioteca COBOL y los mensajes se han instalado correctamente y que LANG y NLSPATH se han especificado correctamente.

Acción del sistema: No se ha realizado ninguna acción del sistema.

IWZ995C

Las variantes de mensaje incluyen:

- ***excepción del sistema* señal recibida al ejecutar la rutina *nombre-rutina* en el desplazamiento 0x*valor-desplazamiento*.**
- **Se ha recibido la señal *excepción del sistema* al ejecutar el código en la ubicación 0x*valor-desplazamiento*.**
- Se ha recibido la señal de ***excepción del sistema* . No se ha podido determinar la ubicación.**

Explicación: El sistema operativo ha detectado una acción no permitida, como un intento de almacenar en un área protegida de memoria o el sistema operativo ha detectado que ha pulsado la tecla de interrupción (normalmente la tecla Control-C, pero se puede volver a configurar).

Respuesta del programador: Si la señal se debe a una acción no permitida, ejecute el programa bajo el depurador y le proporcionará información más precisa sobre dónde se ha producido el error. Un ejemplo de este tipo de error es un puntero con un valor no permitido.

Acción del sistema: La aplicación ha terminado.

IWZ2502S

UTC/GMT no estaba disponible en el sistema.

Explicación: Una llamada a CEEUTC o CEEGMT ha fallado porque el reloj del sistema estaba en un estado no válido. No se ha podido determinar la hora actual.

Respuesta del programador: Notifique al personal de soporte del sistema que el reloj del sistema está en un estado no válido.

Acción del sistema: Todos los valores de salida se establecen en 0.

Código de comentarios simbólicos: CEE2E6

IWZ2503S

El desplazamiento de UTC/GMT a la hora local no estaba disponible desde el sistema.

Explicación: Una llamada a CEEGMTO ha fallado porque (1) no se ha podido determinar el sistema operativo actual o (2) parece que el campo de huso horario del bloque de control del sistema operativo contiene datos no válidos.

Respuesta del programador: Notifique al personal de soporte de sistemas que el desplazamiento de hora local almacenado en el sistema operativo parece contener datos no válidos.

Acción del sistema: Todos los valores de salida se establecen en 0.

Código de comentarios simbólicos: CEE2E7

IWZ2505S

El valor de input_seconds en una llamada a CEEDATM o CEESECI no estaba dentro del rango soportado.

Explicación: El valor de input_seconds pasado en una llamada a CEEDATM o CEESECI no era un número de coma flotante entre 86,400.0 y 265,621,679,999.999 El parámetro de entrada debe representar el número de segundos transcurridos desde 00:00:00 el 14 de octubre de 1582, siendo 00 :00:00.000 15 de octubre de 1582 la primera fecha/hora soportada, y 23 :59:59.999 31 de diciembre de 9999 siendo la última fecha/hora soportada.

Respuesta del programador: Verifique que el parámetro de entrada contiene un valor de coma flotante entre 86,400.0 y 265,621,679,999.999.

Acción del sistema: Para CEEDATM, el valor de salida se establece en blancos. Para CEESECI, todos los parámetros de salida se establecen en 0.

Código de comentarios simbólicos: CEE2E9

IWZ2506S

Se ha utilizado una era (< JJJ>, < CCCC> o < CCCCCCCCCC>) en una serie de imagen pasada a CEEDATM, pero el valor de número de segundos de entrada no estaba dentro del rango soportado. No se pudo determinar la era.

Explicación: En una llamada CEEDATM, la serie de imagen indica que el valor de entrada se va a convertir a una era; sin embargo, el valor de entrada que se ha especificado está fuera del rango de eras soportadas.

Respuesta del programador: Verifique que el valor de entrada contiene un valor válido de número de segundos dentro del rango de eras soportadas.

Acción del sistema: El valor de salida se establece en blancos.

IWZ2507S

Se han pasado datos insuficientes a CEEDAYS o CEESECS. El valor Lilian no se ha calculado.

Explicación: La serie de imagen pasada en una llamada CEEDAYS o CEESECS no contenía suficiente información. Por ejemplo, es un error utilizar la serie de imagen 'MM/DD ' (sólo mes y día) en una llamada a CEEDAYS o CEESECS, porque falta el valor de año. La información mínima necesaria para calcular un valor de Lilian es (1) mes, día y año, o (2) año y día juliano.

Respuesta del programador: Verifique que la serie de imagen especificada en una llamada a CEEDAYS o CEESECS especifica, como mínimo, la ubicación en la serie de entrada de (1) el año, el mes y el día, o (2) el año y el día juliano.

Acción del sistema: El valor de salida se establece en 0.

Código de comentarios simbólicos: CEE2EB

IWZ2508S

El valor de fecha pasado a CEEDAYS o CEESECS no era válido.

Explicación: En una llamada CEEDAYS o CEESECS, el valor del campo DD o DDD no es válido para el año o mes determinado. Por ejemplo, 'MM/DD/AA' con '02/29/90', o 'YYYY.DDD' con '1990.366' no son válidos porque 1990 no es un año bisiesto. Este código también se puede devolver para cualquier valor de fecha no existente como, por ejemplo, junio 31st, enero 0.

Respuesta del programador: Verifique que el formato de los datos de entrada coincide con la especificación de serie de imagen y que los datos de entrada contienen una fecha válida.

Acción del sistema: El valor de salida se establece en 0.

Código de comentarios simbólicos: CEE2EC

IWZ2509S

No se ha reconocido la era pasada a CEEDAYS o CEESECS.

Explicación: El valor del campo < JJJJ>, < CCCC> o < CCCCCCCC > pasado en una llamada a CEEDAYS o CEESECS no contiene un nombre de era soportado.

Respuesta del programador: Verifique que el formato de los datos de entrada coincide con la especificación de serie de imagen y que la ortografía del nombre de era es correcta. Tenga en cuenta que el nombre de era debe ser una serie DBCS adecuada donde '<' posición debe contener el primer byte del nombre de era.

Acción del sistema: El valor de salida se establece en 0.

IWZ2510S

El valor de horas en una llamada a CEEISEC o CEESECS no se ha reconocido.

Explicación: (1) En una llamada CEEISEC, el parámetro de horas no contenía un número entre 0 y 23, o (2) en una llamada CEESECS, el valor del campo HH (horas) no contiene un número entre 0 y 23, o el "AP" (a.m. /p.m.) el campo está presente y el campo HH no contiene un número entre 1 y 12.

Respuesta del programador: Para CEEISEC, verifique que el parámetro de horas contiene un entero entre 0 y 23. Para CEESECS, verifique que el formato de los datos de entrada coincide con la especificación de serie de imagen y que el campo de horas contiene un valor entre 0 y 23 (o 1 y 12 si se utiliza el campo "AP").

Acción del sistema: El valor de salida se establece en 0.

Código de comentarios simbólicos: CEE2EE

IWZ2511S

El parámetro de día pasado en una llamada CEEISEC no era válido para el año y el mes especificados.

Explicación: El parámetro de día pasado en una llamada CEEISEC no contenía un número de día válido. La combinación de año, mes y día ha formado un valor de fecha no válido. Ejemplos: year=1990, month=2, day=29; o month=6, day=31; o day=0.

Respuesta del programador: Verifique que el parámetro de día contiene un entero entre 1 y 31 y que la combinación de año, mes y día representa una fecha válida.

Acción del sistema: El valor de salida se establece en 0.

Código de comentarios simbólicos: CEE2EF

IWZ2512S

El valor de fecha de Lilian pasado en una llamada a CEEDATE o CEEDYWK no estaba dentro del rango soportado.

Explicación: El número de día de Lilian pasado en una llamada a CEEDATE o CEEDYWK no era un número entre 1 y 3.074.324.

Respuesta del programador: Verifique que el parámetro de entrada contiene un entero entre 1 y 3.074.324.

Acción del sistema: El valor de salida se establece en blancos.

Código de comentarios simbólicos: CEE2EG

IWZ2513S

La fecha de entrada pasada en una llamada CEEISEC, CEEDAYS o CEESECS no estaba dentro del rango soportado.

Explicación: La fecha de entrada pasada en una llamada CEEISEC, CEEDAYS o CEESECS era anterior al 15 de octubre de 1582 o posterior al 31 de diciembre de 9999.

Respuesta del programador: Para CEEISEC, verifique que los parámetros de año, mes y día forman una fecha posterior o igual al 15 de octubre de 1582. Para CEEDAYS y CEESECS, verifique que el formato de la fecha de entrada coincide con la especificación de serie de imagen y que la fecha de entrada está dentro del rango soportado.

Acción del sistema: El valor de salida se establece en 0.

Código de comentarios simbólicos: CEE2EH

IWZ2514S

El valor de año pasado en una llamada CEEISEC no estaba dentro del rango soportado.

Explicación: El parámetro de año pasado en una llamada CEEISEC no contenía un número entre 1582 y 9999.

Respuesta del programador: Verifique que el parámetro de año contiene datos válidos y que el parámetro de año incluye el siglo, por ejemplo, especifique el año 1990, no el año 90.

Acción del sistema: El valor de salida se establece en 0.

Código de comentarios simbólicos: CEE2EI

IWZ2515S

El valor de milisegundos en una llamada CEEISEC no se ha reconocido.

Explicación: En una llamada CEEISEC, el parámetro de milisegundos (*input_milisegundos*) no contenía un número entre 0 y 999.

Respuesta del programador: Verifique que el parámetro de milisegundos contiene un entero entre 0 y 999.

Acción del sistema: El valor de salida se establece en 0.

Código de comentarios simbólicos: CEE2EJ

IWZ2516S

No se ha reconocido el valor de minutos en una llamada CEEISEC.

Explicación: (1) En una llamada CEEISEC, el parámetro de minutos (*input_minutes*) no contenía un número entre 0 y 59, o (2) en una llamada CEESECS, el valor del campo MI (minutos) no contenía un número entre 0 y 59.

Respuesta del programador: Para CEEISEC, verifique que el parámetro de minutos contiene un entero entre 0 y 59. Para CEESECS, verifique que el formato de los datos de entrada coincide con la especificación de serie de imagen y que el campo de minutos contiene un número entre 0 y 59.

Acción del sistema: El valor de salida se establece en 0.

Código de comentarios simbólicos: CEE2EK

IWZ2517S

No se ha reconocido el valor de mes en una llamada CEEISEC.

Explicación: (1) En una llamada CEEISEC, el parámetro month (*input_month*) no contenía un número entre 1 y 12, o (2) en una llamada CEEDAYS o CEESECS, el valor del campo MM no contenía un número entre 1 y 12, o el valor de MMM, MMMM, etc. El campo no contenía un nombre de mes o abreviatura de mes correctamente escrito en el idioma nacional activo actualmente.

Respuesta del programador: Para CEEISEC, verifique que el parámetro month contiene un entero entre 1 y 12. Para CEEDAYS y CEESECS, verifique que el formato de los datos de entrada coincide con la especificación de serie de imagen. Para el campo MM, verifique que el valor de entrada esté entre 1 y 12. Para nombres de mes deletreados (MMM, MMMM, etc.), verificar que la ortografía o abreviatura del nombre del mes es correcta en el idioma nacional actualmente activo.

Acción del sistema: El valor de salida se establece en 0.

Código de comentarios simbólicos: CEE2EL

IWZ2518S

Se ha especificado una serie de imagen no válida en una llamada a un servicio de fecha/hora.

Explicación: La serie de imagen proporcionada en una llamada a uno de los servicios de fecha/hora no era válida. Sólo se puede especificar una serie de caracteres de era.

Respuesta del programador: Verifique que la serie de imagen contiene datos válidos. Si la serie de imagen contiene más de un descriptor de era, como por ejemplo < JJJJ > y < CCCC>, cambie la serie de imagen para utilizar sólo una era.

Acción del sistema: El valor de salida se establece en 0.

Código de comentarios simbólicos: CEE2EM

IWZ2519S

No se ha reconocido el valor de segundos en una llamada CEEISEC.

Explicación: (1) En una llamada CEEISEC, el parámetro de segundos (*input_seconds*) no contenía un número entre 0 y 59, o (2) en una llamada CEESECS, el valor del campo SS (segundos) no contenía un número entre 0 y 59.

Respuesta del programador: Para CEEISEC, verifique que el parámetro de segundos contiene un entero entre 0 y 59. Para CEESECS, verifique que el formato de los datos de entrada coincide con la especificación de serie de imagen y que el campo de segundos contiene un número entre 0 y 59.

Acción del sistema: El valor de salida se establece en 0.

Código de comentarios simbólicos: CEE2EN

IWZ2520S

CEEDAYS ha detectado datos no numéricos en un campo numérico, o la serie de fecha no coincidía con la serie de imagen.

Explicación: El valor de entrada pasado en una llamada CEEDAYS no parecía estar en el formato descrito por la especificación de imagen, por ejemplo, aparecen caracteres no numéricos donde sólo se esperan caracteres numéricos.

Respuesta del programador: Verifique que el formato de los datos de entrada coincide con la especificación de serie de imagen y que los campos numéricos sólo contienen datos numéricos.

Acción del sistema: El valor de salida se establece en 0.

Código de comentarios simbólicos: CEE2EO

IWZ2521S

El valor de < JJJJ>, < CCCC> o < CCCCCCCC> year-within-era pasado a CEEDAYS o CEESECS era cero.

Explicación: En una llamada CEEDAYS o CEESECS, si se especifica la señal de imagen YY o ZYY, y si la serie de imagen contiene una de las señales de era como, por ejemplo, < CCCC > o < JJJJ>, el valor del año debe ser mayor o igual que 1 y debe ser un valor de año válido para la era. En este contexto, el campo YY o ZYY significa año dentro de la era.

Respuesta del programador: Verifique que el formato de los datos de entrada coincide con la especificación de serie de imagen y que los datos de entrada son válidos.

Acción del sistema: El valor de salida se establece en 0.

IWZ2522S

Se ha utilizado una era (< JJJJ>, < CCCC> o < CCCCCCCC>) en una serie de imagen pasada a CEEDATE, pero el valor de fecha de Lilian no estaba dentro del rango soportado. No se pudo determinar la era.

Explicación: En una llamada CEEDATE, la serie de imagen indica que la fecha de Lilian se va a convertir en una era, pero la fecha de Lilian está fuera del rango de eras soportadas.

Respuesta del programador: Verifique que el valor de entrada contiene un número de día Lilian válido dentro del rango de eras soportadas.

Acción del sistema: El valor de salida se establece en blancos.

IWZ2525S

CEESECS ha detectado datos no numéricos en un campo numérico, o la serie de indicación de fecha y hora no coincide con la serie de imagen.

Explicación: El valor de entrada pasado en una llamada CEESECS no parecía estar en el formato descrito por la especificación de imagen. Por ejemplo, aparecen caracteres no numéricos donde sólo se esperan caracteres numéricos, o a.m./p.m. campo (AP, A.P., etc.) no contenía las series 'AM' o 'PM'.

Respuesta del programador: Verifique que el formato de los datos de entrada coincide con la especificación de serie de imagen y que los campos numéricos sólo contienen datos numéricos.

Acción del sistema: El valor de salida se establece en 0.

Código de comentarios simbólicos: CEE2ET

IWZ2526S

La serie de fecha devuelta por CEEDATE se ha truncado.

Explicación: En una llamada CEEDATE, la serie de salida no era lo suficientemente grande como para contener el valor de fecha con formato.

Respuesta del programador: Verifique que el elemento de datos de serie de salida sea lo suficientemente grande como para contener toda la fecha formateada. Asegúrese de que el parámetro de salida sea al menos tan largo como el parámetro de serie de imagen.

Acción del sistema: El valor de salida se trunca en la longitud del parámetro de salida.

Código de comentarios simbólicos: CEE2EU

IWZ2527S

La serie de indicación de fecha y hora devuelta por CEEDATM se ha truncado.

Explicación: En una llamada CEEDATM, la serie de salida no era lo suficientemente grande como para contener el valor de indicación de fecha y hora formateado.

Respuesta del programador: Verifique que el elemento de datos de serie de salida sea lo suficientemente grande como para contener toda la indicación de fecha y hora formateada. Asegúrese de que el parámetro de salida sea al menos tan largo como el parámetro de serie de imagen.

Acción del sistema: El valor de salida se trunca en la longitud del parámetro de salida.

Código de comentarios simbólicos: CEE2EV

IWZ2531S

La hora local no estaba disponible desde el sistema.

Explicación: Una llamada a CEELOCT ha fallado porque el reloj del sistema estaba en un estado no válido. No se puede determinar la hora actual.

Respuesta del programador: Notifique al personal de soporte del sistema que el reloj del sistema está en un estado no válido.

Acción del sistema: Todos los valores de salida se establecen en 0.

Código de comentarios simbólicos: CEE2F3

IWZ2533S

El valor pasado a CEESCEN no estaba entre 0 y 100.

Explicación: El valor *century_start* pasado en una llamada CEESCEN no estaba entre 0 y 100, ambos incluidos.

Respuesta del programador: Asegúrese de que el parámetro de entrada está dentro del rango.

Acción del sistema: no se realiza ninguna acción del sistema; la ventana de 100 años que se supone para todos los años de dos dígitos no cambia.

Código de comentarios simbólicos: CEE2F5

IWZ2534W

Se ha especificado una anchura de campo insuficiente para un nombre de mes o día de la semana en una llamada a CEEDATE o CEEDATM. La salida se ha establecido en blancos.

Explicación: los servicios invocables CEEDATE o CEEDATM emite este mensaje siempre que la serie de imagen contiene MMM, MMMMMZ, WWW, Wwww, etc., solicitar un nombre de mes o un nombre de día de la semana, y el nombre de mes que se está formateando actualmente contiene más caracteres de los que pueden caber en el campo indicado.

Respuesta del programador: Aumente el ancho del campo especificando suficientes Ms o W para contener el nombre de mes o día de la semana más largo que se está formateando.

Acción del sistema: los campos de nombre de mes y nombre de día de la semana que son de anchura insuficiente se establecen en espacios en blanco. El resto de la serie de salida no se ve afectada. El proceso continúa.

Código de comentarios simbólicos: CEE2F6

Conceptos relacionados
Apéndice B, “Resultados intermedios y precisión aritmética”, en la página 551

Tareas relacionadas
“Establecimiento de variables de entorno” en la página 229
“Generación de una lista de mensajes del compilador” en la página 246

Avisos

Esta información se ha desarrollado para productos y servicios ofrecidos en EE.UU.

Es posible que IBM no ofrezca los productos, servicios o funciones que se tratan en esta publicación en otros países. Consulte al representante local de IBM para obtener información sobre los productos y servicios disponibles actualmente en su área. Las referencias a programas, productos o servicios de IBM no pretenden establecer ni implicar que sólo puedan utilizarse dichos productos, programas o servicios de IBM. En su lugar, se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no infrinja ninguno de los derechos de propiedad intelectual de IBM. No obstante, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

Es posible que IBM tenga patentes o solicitudes de patente pendientes que traten el tema descrito en este documento. El hecho de proporcionar este documento no concede ninguna licencia sobre estas patentes. Puede realizar consultas sobre licencias escribiendo a:

IBM Director de licencias
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
EE. UU.

Para consultas sobre licencias en las que se solicite información sobre juegos de caracteres de doble byte (DBCS), póngase en contacto con el departamento de propiedad intelectual de IBM de su país o envíe sus consultas, por escrito, a la dirección siguiente:

Licencia de propiedad intelectual
Ley de Propiedad Intelectual y Legal
IBM Japón, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japón

El párrafo siguiente no se aplica al Reino Unido ni a ningún otro país donde estas disposiciones sean incompatibles con la legislación local: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍAS DE NINGÚN TIPO, NI EXPLÍCITAS NI IMPLÍCITAS, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN, COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN DETERMINADO. Algunos países no permiten la renuncia a garantías explícitas o implícitas en ciertas transacciones, por lo que la declaración anterior puede no aplicarse en su caso.

Esta información puede incluir imprecisiones técnicas o errores tipográficos. Periódicamente se realizan cambios en la información aquí contenida; estos cambios se incorporarán en nuevas ediciones de la publicación. IBM puede reservarse el derecho de realizar mejoras y/o cambios en los productos y/o programas descritos en esta publicación en cualquier momento sin previo aviso.

Las referencias contenidas en esta información a sitios web que no son deIBM se proporcionan únicamente para su comodidad y no constituyen en modo alguno un aval de dichos sitios web. Los materiales de dichos sitios web no forman parte de los materiales para este producto IBM y el uso de dichos sitios web es a cuenta y riesgo del usuario.

IBM puede utilizar o distribuir la información que usted le suministre del modo que IBM considere conveniente sin incurrir por ello en ninguna obligación para con usted.

Los titulares de licencias de este programa que deseen obtener información sobre el mismo con el fin de permitir: (i) el intercambio de información entre programas creados independientemente y otros programas (incluido éste) y (ii) el uso mutuo de la información que se ha intercambiado, deben ponerse en contacto con:

IBM Director de licencias
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
EE. UU.

Dicha información puede estar disponible, sujeta a los términos y condiciones correspondientes, incluyendo, en algunos casos, el pago de una tarifa.

El programa bajo licencia descrito en este documento y todo el material bajo licencia disponible para el mismo son proporcionados por IBM bajo los términos del Acuerdo de cliente de IBM , IBM Acuerdo internacional de licencia de programa o cualquier acuerdo equivalente entre las partes.

Todos los datos de rendimiento contenidos en el presente documento se han obtenido en un entorno controlado. Por tanto, los resultados obtenidos en otros entornos operativos pueden variar de forma significativa. Algunas de las medidas podrían proceder de sistemas en proceso de desarrollo y no se garantiza que dichas medidas sean las mismas en sistemas disponibles para uso general. Además, es posible que algunas de las medidas se hayan estimado a través de una extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben consultar los datos que corresponden a su entorno específico.

La información relativa a productos que no son de IBM se obtuvo de los proveedores de esos productos, sus anuncios publicados u otras fuentes de disponibilidad pública. IBM no ha probado estos productos y no puede confirmar la precisión de su rendimiento, compatibilidad o cualquier otro aspecto relacionado con los productos que no son de IBM. Las preguntas relacionadas con productos que no son de IBM deberán dirigirse a los proveedores de estos productos.

Las declaraciones relativas a la dirección o intenciones futuras de IBM pueden cambiar o ser retiradas sin aviso, y representan sólo propósitos y objetivos.

Esta información contiene ejemplos de datos e informes utilizados en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen los nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier parecido con los nombres y direcciones utilizados por una empresa real es mera coincidencia.

LICENCIA DE DERECHOS DE AUTOR:

Esta información contiene programas de aplicación de ejemplo en lenguaje fuente, que ilustra técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo de cualquier forma sin realizar ningún pago a IBM, con el fin de desarrollar, utilizar, comercializar o distribuir programas de aplicación que se ajusten a la interfaz de programación de aplicaciones para la plataforma operativa para la que se han escrito los programas de ejemplo. Estos ejemplos no se han probado a fondo en todas las condiciones. Por lo tanto, IBMno puede garantizar ni dar por supuesta la fiabilidad, la capacidad de servicio o el funcionamiento de estos programas. Los programas de ejemplo se proporcionan "TAL CUAL", sin garantía de ningún tipo. IBM no será responsable de ningún daño resultante del uso de los programas de ejemplo.

Cada copia o parte de estos programas de ejemplo o cualquier trabajo derivado, debe incluir un aviso de copyright como el siguiente:

© (nombre de su empresa) (año). Partes de este código derivan de IBM Corp. Programas de ejemplo. © Copyright IBM Corp. 1995, 2019.

CONSIDERACIONES SOBRE LA POLÍTICA DE PRIVACIDAD:

IBM , incluido el software como soluciones de servicio, ("Ofertas de software") pueden utilizar cookies u otras tecnologías para recopilar información de uso del producto, para ayudar a mejorar la experiencia del usuario final o para adaptar las interacciones con el usuario final, o para otros fines. En muchos casos, las ofertas de software no recopilan información de identificación personal. Algunas de nuestras ofertas de software pueden ayudarle a recopilar información de identificación personal. Si esta oferta de software utiliza cookies para recopilar información de identificación personal, la información específica sobre el uso de cookies de esta oferta se establece a continuación.

Esta Oferta de Software no utiliza cookies u otras tecnologías para recopilar información de identificación personal.

Si las configuraciones desplegadas para esta oferta de software le proporcionan como cliente la capacidad de recopilar información de identificación personal de los usuarios finales a través de cookies y otras tecnologías, debe buscar su propio asesoramiento legal sobre cualquier legislación aplicable a dicha recopilación de datos, incluidos los requisitos de aviso y consentimiento.

Para obtener más información sobre el uso de diversas tecnologías, incluidas las cookies, para estos fines, consulte la Política de privacidad de IBMen http://www.ibm.com/privacy y la Declaración de privacidad en línea de IBMen http://www.ibm.com/privacy/details en la sección titulada "Cookies, Web Beacons y otras tecnologías," y "Declaración de privacidad de productos de software y software como servicio de IBM" en http://www.ibm.com/software/info/product-privacy.

Marcas comerciales

IBM, el logotipo de IBM e ibm.com son marcas registradas de International Business Machines Corp. registradas en muchas jurisdicciones en todo el mundo. Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas. Hay disponible una lista actual de marcas registradas de IBM en la web en "Copyright and trademark information" en www.ibm.com/legal/copytrade.shtml.

Intel es una marca registrada de Intel Corporation o sus filiales en Estados Unidos y otros países.

Java y todas las marcas y logotipos basados en Java son marcas comerciales o marcas registradas de Oracle y de sus filiales.

Linux es una marca registrada de Linus Torvalds en Estados Unidos y/o en otros países.

Microsoft y Windows son marcas registradas de Microsoft Corporation en Estados Unidos y/o en otros países.

UNIX es una marca registrada de The Open Group en Estados Unidos y otros países.

Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas.

Glosario

Los términos de este glosario se definen de acuerdo con su significado en COBOL. Estos términos pueden tener o no el mismo significado en otros idiomas.

glossary.html

Este glosario incluye términos y definiciones de las publicaciones siguientes:

- *ANSI INCITS 23-1985, Lenguajes de programación-COBOL*, modificado por *ANSI INCITS 23a-1989, Idiomas de programación-COBOL-Módulo de función intrínseca para COBOL*, y *ANSI INCITS 23b-1993, Idiomas de programación-Enmienda de corrección para COBOL*
- *ISO 1989:1985, Lenguajes de programación-COBOL*, modificado por *ISO/IEC 1989/AMD1:1992, Lenguajes de programación-COBOL: Módulo de función intrínseca*
- *ANSI X3.172-2002, American National Standard Dictionary for Information Systems*
- *INCITS/ISO/IEC 1989-2002, Tecnologías de la información-Lenguajes de programación-COBOL*
- *INCITS/ISO/IEC 1989:2014, Tecnologías de la información-Lenguajes de programación, sus entornos e interfaces de software del sistema-Lenguaje de programación COBOL*

Las definiciones de American National Standard van precedidas de un asterisco (*).

A

*** condición de relación combinada abreviada**
La condición combinada que resulta de la omisión explícita de un sujeto común o un sujeto común y operador relacional común en una secuencia consecutiva de condiciones de relación.

terminación anómala
Terminación anómala de un programa.

*** modalidad de acceso**
La forma en la que se van a operar los registros dentro de un archivo.

*** coma decimal real**
La representación física, utilizando los caracteres de coma decimal punto (.) o coma (,), de la posición de coma decimal en un elemento de datos.

codificación de documento real
Para un documento XML, una de las siguientes categorías de codificación que el analizador XML determina examinando los primeros bytes del documento:

- ASCII
- EBCDIC
- UTF-8
- UTF-16, ya sea big-endian o little-endian
- Otra codificación no soportada
- Sin codificación reconocible

Linux sistema de archivos nativo
Cualquiera de los sistemas de archivos locales o de red que soportan directamente archivos continuos codificados o binarios.

Los sistemas de archivos nativos Linux dan soporte a archivos secuenciales de línea directamente y se utilizan como almacén de archivos para todos los demás tipos de archivos COBOL.

*** nombre-alfabeto**
Palabra definida por el usuario, en el párrafo `SPECIAL-NAMES` de `ENVIRONMENT DIVISION`, que asigna un nombre a un juego de caracteres específico o a una secuencia de clasificación o a ambos.

*** carácter alfabético**
Una letra o un carácter de espacio.

elemento de datos alfabéticos
Elemento de datos que se describe con una serie de caracteres PICTURE que contiene sólo el símbolo A. Un elemento de datos alfabético tiene USAGE DISPLAY.

*** carácter alfanumérico**
Cualquier carácter del juego de caracteres de un solo byte del sistema.

posición de carácter alfanumérico
Véase *posición de carácter*.

elemento de datos alfanuméricos
Referencia general a un elemento de datos que se describe implícita o explícitamente como USAGE DISPLAY, y que tiene una categoría alfanumérica, alfanumérica editada o numérica editada.

elemento de datos editado alfanumérico
Elemento de datos que se describe mediante una serie de caracteres PICTURE que contiene al menos una instancia del símbolo A o X y al menos uno de los símbolos de inserción simples B, 0o /. Un elemento de datos editado alfanumérico tiene USAGE DISPLAY.

*** función alfanumérica**
Función cuyo valor se compone de una serie de uno o más caracteres del conjunto de caracteres alfanuméricos del sistema.

elemento de grupo alfanumérico
Elemento de grupo que se define sin una cláusula GROUP-USAGE NATIONAL . Para operaciones como INSPECT, STRINGy UNSTRING, un elemento de grupo alfanumérico se procesa como si todo su contenido se describiera como USAGE DISPLAY independientemente del contenido real del grupo. Para las operaciones que requieren el proceso de los elementos elementales dentro de un grupo, como MOVE CORRESPONDING, ADD CORRESPONDINGo INITIALIZE, un elemento de grupo alfanumérico se procesa utilizando la semántica de grupo.

LITERAL ALFANUMÉRICO
Un literal que tiene un delimitador de apertura del conjunto siguiente: ', ", X', X", Z'o Z". La serie de caracteres puede incluir cualquier carácter en el juego de caracteres del sistema.

*** clave de registro alternativa**
Clave, distinta de la clave de registro principal, cuyo contenido identifica un registro dentro de un archivo indexado.

ANSI (American National Standards Institute)
Organización que consta de productores, consumidores y grupos de interés general y establece los procedimientos mediante los cuales las organizaciones acreditadas crean y mantienen estándares voluntarios de la industria en los Estados Unidos.

argumento
(1) Un identificador, un literal, una expresión aritmética o un identificador de función que especifica un valor que debe utilizarse en la evaluación de una función. (2) Un operando de la frase USING de una sentencia CALL , utilizada para pasar valores a un programa llamado.

*** operación aritmética**
El proceso causado por la ejecución de una sentencia aritmética, o la evaluación de una expresión aritmética, que da como resultado una solución matemáticamente correcta a los argumentos presentados.

*** operador aritmético**
Un único carácter, o una combinación fija de dos caracteres que pertenece al siguiente conjunto:

Carácter	Significado
+	Suma
-	Resta
*	Multiplicación

Carácter	Significado
/	División
**	Exponenciación

*** sentencia aritmética**
Sentencia que hace que se ejecute una operación aritmética. Las sentencias aritméticas son ADD, COMPUTE, DIVIDE, MULTIPLYy SUBTRACT.

matriz
Agregado que consta de objetos de datos, a cada uno de los cuales se puede hacer referencia de forma exclusiva mediante subscripción. Una matriz es aproximadamente análoga a una tabla COBOL.

*** clave ascendente**
Clave sobre los valores de los que se ordenan los datos, empezando por el valor más bajo de la clave hasta el valor más alto de la clave, de acuerdo con las reglas para comparar elementos de datos.

ASCII
American National Standard Code for Information Interchange. El código estándar utiliza un juego de caracteres codificado que se basa en caracteres codificados de 7 bits (8 bits incluyendo comprobación de paridad). El estándar se utiliza para el intercambio de información entre sistemas de procesamiento de datos, sistemas de comunicación de datos y equipo asociado. El conjunto ASCII consta de caracteres de control y caracteres gráficos.

IBM ha definido una extensión para ASCII (caracteres 128-255).

Página de códigos multibyte basada en ASCII
Una página de códigos UTF-8, EUC o ASCII DBCS. Cada página de códigos multibyte basada en ASCII incluye caracteres de un solo byte y de varios bytes. La codificación de los caracteres de un solo byte es la codificación ASCII.

DBCS ASCII
Véase *ASCII de doble byte*.

nombre-asignación
Nombre que identifica la organización de un archivo COBOL y el nombre por el que el sistema lo conoce.

*** punto decimal asumido**
Posición de coma decimal que no implica la existencia de un carácter real en un elemento de datos. La coma decimal asumida tiene un significado lógico pero no tiene representación física.

AT END condición
Una condición que se produce durante la ejecución de una sentencia READ, RETURNo SEARCH bajo determinadas condiciones:

- Una sentencia READ se ejecuta en un archivo al que se accede secuencialmente cuando no existe ningún siguiente registro lógico en el archivo, o cuando el número de dígitos significativos en el número de registro relativo es mayor que el tamaño del elemento de datos de clave relativa, o cuando no está disponible un archivo de entrada opcional.
- Una sentencia RETURN se ejecuta cuando no existe ningún registro lógico siguiente para el archivo de clasificación o fusión asociado.
- Una sentencia SEARCH se ejecuta cuando la operación de búsqueda termina sin satisfacer la condición especificada en ninguna de las frases WHEN asociadas.

B

juego de caracteres básico
El conjunto básico de caracteres utilizados en la escritura de palabras, series de caracteres y separadores del idioma. El juego de caracteres básico se implementa en caracteresde un solo byte. El juego de caracteres ampliado incluye los caracteres DBCS, UTF-8o EUC , que se pueden utilizar en comentarios, literales y palabras definidas por el usuario.

Sinónimo de *juego de caracteres COBOL* en 85 COBOL Estándar.

big-endian
El formato predeterminado que el sistema principal y la estación de trabajo de Linux utilizan para almacenar datos binarios y caracteres UTF-16 . En este formato, el byte menos significativo de un elemento de datos binarios está en la dirección más alta y el byte menos significativo de un carácter UTF-16 está en la dirección más alta. Compare con *little-endian*.

elemento binario
Elemento de datos numérico que se representa en notación binaria (en el sistema de numeración base 2). El equivalente decimal consta de los dígitos decimales del 0 al 9, más un signo operativo. El bit situado más a la izquierda del elemento es el signo operativo.

búsqueda binaria
Una búsqueda de dicotomización en la que, en cada paso de la búsqueda, el conjunto de elementos de datos se divide por dos; se realiza una acción adecuada en el caso de un número impar.

*** bloque**
Unidad física de datos que normalmente se compone de uno o más registros lógicos. Para archivos de almacenamiento masivo, un bloque puede contener una parte de un registro lógico. El tamaño de un bloque no tiene relación directa con el tamaño del archivo en el que está contenido el bloque o con el tamaño de los registros lógicos que están contenidos en el bloque o que solapan el bloque. Sinónimo de *registro físico*.

condición booleana
Una condición booleana determina si un literal booleano es verdadero o falso. Una condición booleana sólo se puede utilizar en una expresión condicional constante.

literal booleano
Puede ser B '1', que indica un valor verdadero, o B '0', que indica un valor falso. Los literales booleanos sólo se pueden utilizar en expresiones condicionales constantes.

punto de interrupción
Lugar de un programa informático, normalmente especificado por una instrucción, donde la intervención externa o un programa supervisor puede interrumpir el programa a medida que se ejecuta.

almacenamiento intermedio
Parte del almacenamiento que se utiliza para contener datos de entrada o salida temporalmente.

Función incorporada
Véase *función intrínseca*.

byte
Serie que consta de un determinado número de bits, normalmente ocho, tratados como una unidad y que representa un carácter o una función de control.

marca de orden de bytes (BOM)
Carácter Unicode que se puede utilizar al principio del texto UTF-16 o UTF-32 para indicar el orden de bytes del texto posterior; el orden de bytes puede ser big-endian o little-endian.

código de bytes
Código independiente de la máquina generado por el compilador Java y ejecutado por el intérprete Java . (Oracle)

C

programa llamado
Programa que es el objeto de una sentencia CALL . En tiempo de ejecución, el programa llamado y el programa de llamada se combinan para producir una *unidad de ejecución*.

*** programa de llamada**
Programa que ejecuta un CALL en otro programa.

estructura de caso
Lógica de proceso de programa en la que se prueba una serie de condiciones para elegir entre varias acciones resultantes.

CCSID
Consulte *identificador de juego de caracteres codificados*.

Ventana de siglo
Intervalo de 100 años dentro del cual cualquier año de dos dígitos es exclusivo. Hay varios tipos de ventana de siglo disponibles para los programadores de COBOL:

- Para los campos de fecha con ventana, utilice la opción de compilador YEARWINDOW .
- Para las funciones intrínsecas de ventana DATE-TO-YYYYMMDD, DAY-TO-YYYYDDDy YEAR-TO-YYYY, especifique la ventana de siglo con *argument-2*.

*** carácter**
Unidad indivisible básica del idioma.

unidad de codificación de caracteres
Unidad de datos que corresponde a un elemento de código de un juego de caracteres codificado. Se utilizan una o más unidades de codificación de caracteres para representar un carácter en un juego de caracteres codificado. También se conoce como *unidad de codificación*.

Para USAGE NATIONAL, una unidad de codificación de caracteres corresponde a un punto de código de 2 bytes de UTF-16.

Para USAGE DISPLAY, una unidad de codificación de caracteres corresponde a un byte.

Para USAGE DISPLAY-1, una unidad de codificación de caracteres corresponde a un elemento de código de 2 bytes en el juego de caracteres DBCS.

posición de carácter
Cantidad de almacenamiento físico o espacio de presentación necesario para contener o presentar un carácter. El término se aplica a cualquier clase de carácter. Para clases específicas de caracteres, se aplican los términos siguientes:

- *Posición de carácter alfanumérico*, para los caracteres representados en USAGE DISPLAY
- *Posición de carácter DBCS*, para caracteres DBCS representados en USAGE DISPLAY-1
- *Posición de carácter nacional*, para los caracteres representados en USAGE NATIONAL; sinónimo de *unidad de codificación de caracteres* para UTF-16

juego de caracteres
Colección de elementos que se utilizan para representar información textual, pero para los que no se presupone ninguna representación codificada. Véase también *juego de caracteres codificado*.

serie de caracteres
Secuencia de caracteres contiguos que forman una palabra COBOL, un literal, una serie de caracteres PICTURE o una entrada de comentario. Una serie de caracteres debe estar delimitada por separadores.

punto de comprobación
Punto en el que se puede registrar información sobre el estado de un trabajo y el sistema para que el paso de trabajo se pueda reiniciar más adelante.

*** clase**
Entidad que define el comportamiento común y la implementación para cero, uno o más objetos. Los objetos que comparten la misma implementación se consideran objetos de la misma clase. Las clases se pueden definir jerárquicamente, lo que permite que una clase herede de otra.

*** condición de clase**
La proposición (para la que se puede determinar un valor de verdad) de que el contenido de un elemento es totalmente alfabético, es totalmente numérico, es totalmente DBCS, es totalmente Kanji, o consiste exclusivamente en los caracteres que se listan en la definición de un nombre de clase.

*** nombre-clase (de datos)**
Una palabra definida por el usuario que está definida en el párrafo SPECIAL-NAMES del ENVIRONMENT DIVISION; esta palabra asigna un nombre a la proposición (para la que se puede definir un valor de verdad) de que el contenido de un elemento de datos consta exclusivamente de los caracteres que se listan en la definición del nombre de clase.

*** cláusula**
Conjunto ordenado de series de caracteres COBOL consecutivas cuya finalidad es especificar un atributo de una entrada.

Conjunto de caracteres COBOL
El conjunto de caracteres utilizados en la escritura de sintaxis COBOL. El juego de caracteres COBOL completo consta de estos caracteres:

Carácter	Significado
0,1, . . . ,9	Dígito
A, B,... , Z	Letra mayúscula
a, b,... , z	Letra minúscula
	Espacio
+	Signo más
-	Signo menos (guión)
*	Asterisco
/	Inclinado (barra inclinada)
=	Signo igual
$	Símbolo de moneda
,	Coma
;	Punto y coma
.	Punto (coma decimal, punto completo)
"	comilla
'	Apóstrofo
(	Paréntesis izquierdo
)	Paréntesis derecho
>	Mayor que
<	Menor que
:	Dos puntos
_	Subrayado

*** Palabra COBOL**
Véase *palabra*.

Página de códigos
Asignación de caracteres gráficos y significados de función de control a todos los elementos de código. Por ejemplo, una página de códigos podría asignar caracteres y significados a 256 puntos de código para código de 8 bits, y otra página de códigos podría asignar caracteres y significados a 128 puntos de código para código de 7 bits. Por ejemplo, una de las páginas de códigos de IBM para inglés en la estación de trabajo es IBM-1252 y en el host es IBM-1047.

elemento de código
Patrón de bits exclusivo que se define en un juego de caracteres codificado (página de códigos). Los símbolos gráficos y los caracteres de control se asignan a los elementos de código.

juego de caracteres codificado
Conjunto de reglas inequívocas que establecen un juego de caracteres y la relación entre los caracteres del conjunto y su representación codificada. Ejemplos de juegos de caracteres codificados son los juegos de caracteres representados por páginas de códigos ASCII o EBCDIC o por el esquema de codificación UTF-16 para Unicode.

identificador de juego de caracteres codificados (CCSID)
Un número definido por IBMen el rango de 1 a 65.535 que identifica una página de códigos específica.

*** orden de clasificación**
Secuencia en la que los caracteres que son aceptables para un sistema se ordenan para fines de clasificación, fusión, comparación y para procesar archivos indexados secuencialmente.

*** columna**
Posición de byte dentro de una línea de impresión o dentro de una línea de formato de referencia. Las columnas se numeran desde 1, por 1, empezando por la posición más a la izquierda de la línea y extendiéndose hasta la posición más a la derecha de la línea. Una columna contiene un carácter de un solo byte.

*** condición combinada**
Condición que es el resultado de conectar dos o más condiciones con el operador lógico AND o OR . Véase también *condición* y *condición combinada negada*.

*** entrada-comentario**
Una entrada en IDENTIFICATION DIVISION que se utiliza para la documentación y no tiene ningún efecto en la ejecución.

línea de comentario
Una línea de programa de origen representada por un asterisco (*) en el área de indicador de la línea o por un asterisco seguido por un signo mayor que (*>) como la primera serie de caracteres en el área de texto del programa (Área A más Área B), y cualquier carácter del juego de caracteres del sistema que sigue a en el Área A y el Área B de esa línea. Una línea de comentario sólo sirve para la documentación. Una forma especial de línea de comentario representada por una barra inclinada (/) en el área de indicador de la línea y cualquier carácter del juego de caracteres del sistema en el Área A y el Área B de esa línea provoca la expulsión de la página antes de que se imprima el comentario.

*** programa común**
Un programa que, a pesar de estar directamente contenido dentro de otro programa, puede ser llamado desde cualquier programa directa o indirectamente contenido en ese otro programa.

campo de fecha compatible
El significado del término *compatible*, cuando se aplica a campos de fecha, depende de la división COBOL en la que se produce el uso:

- DATA DIVISION: dos campos de fecha son compatibles si tienen USAGE idénticos y cumplen al menos una de las condiciones siguientes:
  - Tienen el mismo formato de fecha.
  - Ambos son campos de fecha con ventana, donde uno solo consta de un año con ventana, DATE FORMAT YY.
  - Ambos son campos de fecha expandidos, donde uno solo consta de un año expandido, DATE FORMAT YYYY.
  - Uno tiene DATE FORMAT YYXXXXy el otro tiene YYXX.
  - Uno tiene DATE FORMAT YYYYXXXXy el otro tiene YYYYXX.

  Un campo de fecha con ventana puede estar subordinado a un elemento de datos que es un grupo de fechas expandido. Los dos campos de fecha son compatibles si el campo de fecha subordinado tiene USAGE DISPLAY, empieza dos bytes después del inicio del campo de fecha expandida de grupo y los dos campos cumplen al menos una de las condiciones siguientes:
  - El campo de fecha subordinado tiene un patrón DATE FORMAT con el mismo número de X que el patrón DATE FORMAT del campo de fecha de grupo.
  - El campo de fecha subordinada tiene DATE FORMAT YY.
  - El campo de fecha de grupo tiene DATE FORMAT YYYYXXXX y el campo de fecha subordinado tiene DATE FORMAT YYXX.
- PROCEDURE DIVISION: Dos campos de fecha son compatibles si tienen el mismo formato de fecha excepto para la parte del año, que se puede incluir en ventanas o expandirse. Por ejemplo, un campo de fecha con ventana con DATE FORMAT YYXXX es compatible con:

- Otro campo de fecha con ventana con DATE FORMAT YYXXX
- Un campo de fecha expandido con DATE FORMAT YYYYXXX

*** compilar**
(1) Para traducir un programa expresado en un lenguaje de alto nivel a un programa expresado en un lenguaje intermedio, lenguaje ensamblador, o un lenguaje de computadora. (2) Preparar un programa de lenguaje de máquina a partir de un programa de computadora escrito en otro lenguaje de programación haciendo uso de la estructura lógica general del programa, o generando más de una instrucción de computadora para cada declaración simbólica, o ambas, así como realizando la función de un ensamblador.

variable de compilación
Un nombre simbólico para un valor literal determinado o el valor de una expresión aritmética de tiempo de compilación tal como especifica la directiva DEFINE o la opción de compilador DEFINE .

*** tiempo de compilación**
La hora a la que el código fuente COBOL se convierte, mediante un compilador COBOL, en un programa de objeto COBOL.

expresión aritmética en tiempo de compilación
Subconjunto de expresiones aritméticas que se especifican en las directivas DEFINE y EVALUATE o en una expresión condicional constante. La diferencia entre las expresiones aritméticas en tiempo de compilación y las expresiones aritméticas regulares es que en una expresión aritmética en tiempo de compilación:

- No se especificará el operador de exponenciación.
- Todos los operandos serán literales numéricos enteros o expresiones aritméticas en las que todos los operandos son literales numéricos enteros.
- La expresión se especificará de tal manera que no se produzca una división por cero.

compilador
Programa que convierte el código fuente escrito en un lenguaje de nivel superior en un código de objeto de lenguaje de máquina.

sentencia de direccionamiento de compilador
Sentencia que hace que el compilador realice una acción específica durante la compilación. Las sentencias de direccionamiento de compilador estándar son COPY, REPLACEy USE.

Directiva de compilador
Directiva que hace que el compilador realice una acción específica durante la compilación. COBOL para Linux da soporte a la directiva de compilador CALLINTERFACE , así como a las directivas de compilador de compilación condicional (DEFINE, EVALUATE e IF).

*** condición compleja**
Condición en la que uno o más operadores lógicos actúan sobre una o más condiciones. Véase también *condición*, *condición simple negada*y *condición combinada negada*.

ODO complejo
Ciertas formas de la cláusula OCCURS DEPENDING ON :

- Elemento o grupo de ubicación variable: un elemento de datos descrito por una cláusula OCCURS con la opción DEPENDING ON va seguido de un elemento o grupo de datos no subordinado. El grupo puede ser un grupo alfanumérico o un grupo nacional.
- Tabla de ubicación variable: un elemento de datos descrito por una cláusula OCCURS con la opción DEPENDING ON va seguido de un elemento de datos no subordinado descrito por una cláusula OCCURS .
- Tabla con elementos de longitud variable: un elemento de datos descrito por una cláusula OCCURS contiene un elemento de datos subordinado descrito por una cláusula OCCURS con la opción DEPENDING ON .
- Nombre de índice para una tabla con elementos de longitud variable.
- Elemento de una tabla con elementos de longitud variable.

Componente
(1) Una agrupación funcional de archivos relacionados. (2) En programación orientada a objetos, un objeto o programa reutilizable que realiza una función específica y está diseñado para trabajar con otros componentes y aplicaciones. JavaBeans es la arquitectura de Oraclepara crear componentes.

*** nombre-sistema**
Nombre del sistema que identifica el sistema en el que se va a compilar o ejecutar el programa.

condición (excepción)
Cualquier alteración del flujo normal programado de una aplicación. Las condiciones pueden ser detectadas por el hardware o el sistema operativo y dar como resultado una interrupción. También se pueden detectar mediante código generado específico del idioma o código de biblioteca de idioma.

condición (expresión)
Estado de los datos en tiempo de ejecución para los que se puede determinar un valor de verdad. Donde se utiliza en esta información en o en referencia a "condición" (*condition-1*, *condition-2*,.). de un formato general, el término se refiere a una expresión condicional que consiste en una condición simple opcionalmente entre paréntesis o una condición combinada (que consiste en la combinación sintácticamente correcta de condiciones simples, operadores lógicos y paréntesis) para los que se puede determinar un valor de verdad. Véase también *condición simple*, *condición compleja*, *condición simple negada*, *condición combinada*, y *condición combinada negada*.

*** expresión condicional**
Una condición simple o una condición compleja especificada en una sentencia EVALUATE, IF, PERFORMo SEARCH . Véase también *condición simple* y *condición compleja*.

*** frase condicional**
Frase que especifica la acción que se debe realizar al determinar el valor de verdad de una condición que resulta de la ejecución de una sentencia condicional.

*** sentencia condicional**
Sentencia que especifica que el valor de verdad de una condición debe determinarse y que la acción subsiguiente del programa objeto depende de este valor de verdad.

*** variable condicional**
Un elemento de datos uno o más valores de los cuales tiene asignado un nombre de condición.

*** nombre-condición**
Palabra definida por el usuario que asigna un nombre a un subconjunto de valores que una variable condicional puede asumir; o una palabra definida por el usuario asignada a un estado de un conmutador o dispositivo definido por el implementador.

*** condición de nombre-condición**
La proposición (para la cual se puede determinar un valor de verdad) de que el valor de una variable condicional es un miembro del conjunto de valores atribuidos a un nombre-condición asociado con la variable condicional.

*** `CONFIGURATION SECTION`**
Una sección de `ENVIRONMENT DIVISION` que describe las especificaciones generales de los programas de origen y objeto.

CONSOLE
Un nombre de entorno COBOL asociado a la consola del operador.

expresión condicional constante
Subconjunto de expresiones condicionales que se pueden utilizar en `IF` directivas o `WHEN` frases de las directivas `EVALUATE` .

Una expresión condicional constante será uno de los siguientes elementos:

- Condición de relación en la que ambos operandos son literales o expresiones aritméticas que sólo contienen términos literales. La condición se regirá por las normas relativas a las condiciones de relación, con las siguientes adiciones:
  - Los operandos serán de la misma categoría. Una expresión aritmética es de la categoría numérica.

– Si se especifican literales y no son literales numéricos, el operador relacional será "IS EQUAL TO", "IS NOT EQUAL TO", "IS =", "IS NOT =" o "IS <>".

Véase también *condición de relación*.

- Una condición definida. Véase también *condición definida*.
- Una condición booleana. Véase también *condición booleana*.
- Una condición compleja formada combinando las formas anteriores de condiciones simples en condiciones complejas utilizando AND, OR y NOT. No se especificarán las condiciones de relación combinadas abreviadas. Véase también *condición compleja*.

programa contenido
Un programa COBOL que está anidado dentro de otro programa COBOL.

*** elementos contiguos**
Elementos que se describen mediante entradas consecutivas en DATA DIVISIONy que tienen una relación jerárquica definida entre sí.

libro de copias
Archivo o miembro de biblioteca que contiene una secuencia de código que se incluye en el programa fuente durante la compilación utilizando la sentencia COPY . El archivo puede ser creado por el usuario, proporcionado por COBOL, o proporcionado por otro producto. Sinónimo de *archivo de copia*.

*** contador**
Elemento de datos utilizado para almacenar números o representaciones de números de una manera que permite que estos números se incrementen o disminuyan por el valor de otro número, o que se cambien o restablezcan a cero o a un valor positivo o negativo arbitrario.

Listado de referencias cruzadas
Parte del listado del compilador que contiene información sobre dónde se definen, hacen referencia y modifican los archivos, los campos y los indicadores en un programa.

valor de signo de moneda
Serie de caracteres que identifica las unidades monetarias almacenadas en un elemento editado numérica-. Los ejemplos típicos son $, USD y EUR. Un valor de signo de moneda se puede definir mediante la opción de compilador CURRENCY o la cláusula CURRENCY SIGN en el párrafo SPECIAL-NAMES de ENVIRONMENT DIVISION. Si no se especifica la cláusula CURRENCY SIGN y la opción de compilador NOCURRENCY está en vigor, se utiliza el signo de dólar ($) como valor de signo de moneda predeterminado. Véase también *símbolo de moneda*.

Símbolo de moneda
Carácter utilizado en una cláusula PICTURE para indicar la posición de un valor de signo de moneda en un elemento editado numérica-editado. Un símbolo de moneda se puede definir mediante la opción de compilador CURRENCY o la cláusula CURRENCY SIGN en el párrafo SPECIAL-NAMES de ENVIRONMENT DIVISION. Si no se especifica la cláusula CURRENCY SIGN y la opción de compilador NOCURRENCY está en vigor, se utiliza el signo de dólar ($) como valor de signo de moneda predeterminado y símbolo de moneda. Se pueden definir varios símbolos de moneda y valores de signo de moneda. Véase también *valor de signo de moneda*.

*** registro actual**
En el proceso de archivos, el registro que está disponible en el área de registro asociada a un archivo.

*** puntero de volumen actual**
Entidad conceptual que apunta al volumen actual de un archivo secuencial.

D

*** cláusula de datos**
Cláusula, que aparece en una entrada de descripción de datos en el DATA DIVISION de un programa COBOL, que proporciona información que describe un atributo determinado de un elemento de datos.

*** entrada de descripción de datos**
Una entrada en el DATA DIVISION de un programa COBOL que se compone de un número de nivel seguido de un nombre de datos, si es necesario, y seguido de un conjunto de cláusulas de datos, según sea necesario.

DATA DIVISION
La división de un programa COBOL que describe los datos que debe procesar el programa: los archivos que se van a utilizar y los registros contenidos en ellos; los registros WORKING-STORAGE internos que serán necesarios; los datos que estarán disponibles en más de un programa en la unidad de ejecución COBOL.

*** elemento de datos**
Unidad de datos (excluyendo literales) definida por un programa COBOL o por las reglas para la evaluación de funciones.

*** nombre-datos**
Palabra definida por el usuario que nombra un elemento de datos descrito en una entrada de descripción de datos. Cuando se utiliza en los formatos generales, el nombre de datos representa una palabra que no debe ser modificada por referencia, subclasificada o calificada a menos que lo permitan específicamente las reglas para el formato.

campo de fecha
Cualquiera de los elementos siguientes:

- Elemento de datos cuya entrada de descripción de datos incluye una cláusula DATE FORMAT .
- Un valor devuelto por una de las siguientes funciones intrínsecas:

```
DATE-OF-INTEGER
DATE-TO-YYYYMMDD
DATEVAL
DAY-OF-INTEGER
DAY-TO-YYYYDDD
YEAR-TO-YYYY
YEARWINDOW
```

- Los elementos de datos conceptuales DATE, DATE YYYYMMDD, DAY y DAY YYYYDDD de la sentencia ACCEPT .
- El resultado de determinadas operaciones aritméticas. Para obtener detalles, consulte Aritmética con campos de fecha (*COBOL for Linux en x86 Consulta de lenguaje*).

El término *campo de fecha* hace referencia tanto al *campo de fecha expandida* como al *campo de fecha con ventana*. Véase también *no fecha*.

formato de fecha
El patrón de fecha de un campo de fecha, especificado de cualquiera de las formas siguientes:

- Explícitamente, mediante la cláusula DATE FORMAT o la función intrínseca DATEVAL argument-2
- Implícitamente, por sentencias y funciones intrínsecas que devuelven campos de fecha. Para obtener detalles, consulte Campo de fecha (*COBOL for Linux en x86 Consulta de lenguaje*).

Sistema de archivosDb2
El sistema de archivos Db2 admite archivos secuenciales, indexados y relativos. Proporciona una interoperación mejorada con CICS, lo que permite a los programas COBOL por lotes acceder a los archivos CICS ESDS, KSDS y RRDS almacenados en Db2.

DBCS
Véase *juego de caracteres de doble byte (DBCS)*.

Carácter DBCS
Cualquier carácter definido en el juego de caracteres de doble byte de IBM.

Posición de carácter DBCS
Véase *posición de carácter*.

elemento de datos DBCS
Un elemento de datos que se describe mediante una serie de caracteres PICTURE que contiene al menos un símbolo Go, cuando la opción de compilador NSYMBOL(DBCS) está en vigor, al menos un símbolo N. Un elemento de datos DBCS tiene USAGE DISPLAY-1.

*** línea de depuración**
Cualquier línea con una D en el área de indicador de la línea.

*** sección de depuración**
Una sección que contiene una sentencia `USE FOR DEBUGGING` .

*** frase declarativa**
Una sentencia de direccionamiento de compilador que consta de una única sentencia USE terminada por el punto separador.

*** declarativas**
Un conjunto de una o más secciones de propósito especial, escritas al principio de `PROCEDURE DIVISION`, la primera de las cuales va precedida de la palabra clave `DECLARATIVE` y la última de las cuales va seguida de las palabras clave `END DECLARATIVES`. Una declarativa se compone de una cabecera de sección, seguida de una frase de dirección del compilador USE , seguida de un conjunto de cero, uno o más párrafos asociados.

*** deseditar**
Eliminación lógica de todos los caracteres de edición de un elemento de datos editado numérico para determinar el valor numérico no editado del elemento.

condición definida
Condición de tiempo de compilación que comprueba si se ha definido una variable de compilación. Las condiciones definidas se especifican en `IF` directivas o `WHEN` frases de las directivas `EVALUATE` .

*** sentencia de ámbito delimitado**
Cualquier sentencia que incluya su terminador de ámbito explícito.

*** delimitador**
Carácter o secuencia de caracteres contiguos que identifican el final de una serie de caracteres y separan dicha serie de caracteres de la siguiente serie de caracteres. Un delimitador no forma parte de la serie de caracteres que delimita.

*** clave descendente**
Clave sobre cuyos valores se ordenan los datos empezando por el valor más alto de clave hasta el valor más bajo de clave, de acuerdo con las reglas para comparar elementos de datos.

dígito
Cualquiera de los números del 0 al 9. En COBOL, el término no se utiliza para hacer referencia a ningún otro símbolo.

*** posición de dígito**
La cantidad de almacenamiento físico necesaria para almacenar un único dígito. Esta cantidad puede variar en función del uso especificado en la entrada de descripción de datos que define el elemento de datos.

*** acceso directo**
El recurso para obtener datos de dispositivos de almacenamiento o para introducir datos en un dispositivo de almacenamiento de tal manera que el proceso dependa solamente de la ubicación de esos datos y no de una referencia a los datos a los que se ha accedido previamente.

visualizar elemento de datos de coma flotante
Elemento de datos que se describe implícita o explícitamente como `USAGE DISPLAY` y que tiene una serie de caracteres `PICTURE` que describe un elemento de datos de coma flotante externo.

*** división**
Colección de cero, una o más secciones o párrafos, denominada cuerpo de división, que se forman y combinan de acuerdo con un conjunto específico de reglas. Cada división consta de la cabecera de división y el cuerpo de división relacionado. Hay cuatro divisiones en un programa COBOL: Identificación, Entorno, Datos y Procedimiento.

*** cabecera de división**
Combinación de palabras seguida de un punto de separación que indica el principio de una división. Las cabeceras de división son:

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
```

```
DATA DIVISION.
PROCEDURE DIVISION.
```

realizar construcción
En la programación estructurada, se utiliza una sentencia DO para agrupar una serie de sentencias en un procedimiento. En COBOL, una sentencia PERFORM en línea funciona de la misma forma.

hacer hasta
En la programación estructurada, se ejecutará un bucle do-until al menos una vez, y hasta que se cumpla una condición determinada. En COBOL, una frase TEST AFTER utilizada con la sentencia PERFORM funciona de la misma forma.

hacer mientras
En la programación estructurada, se ejecutará un bucle do-while si, y mientras, se cumple una condición determinada. En COBOL, una frase TEST BEFORE utilizada con la sentencia PERFORM funciona de la misma forma.

declaración de tipo de documento
Elemento XML que contiene o apunta a declaraciones de marcación que proporcionan una gramática para una clase de documentos. Esta gramática se conoce como definición de tipo de documento o DTD.

definición de tipo de documento (DTD)
Gramática de una clase de documentos XML. Véase *declaración de tipo de documento*.

ASCII de doble byte
Un juego de caracteres IBM que incluye caracteres DBCS y ASCII de un solo byte. (También conocido como DBCS ASCII.)

EBCDIC de doble byte
Un juego de caracteres IBM que incluye caracteres DBCS y EBCDIC de un solo byte. (También conocido como EBCDIC DBCS.)

juego de caracteres de doble byte (DBCS)
Conjunto de caracteres en el que cada carácter está representado por 2 bytes. Los idiomas como el japonés, el chino y el coreano, que contienen más símbolos de los que pueden representarse mediante 256 elementos de código, requieren juegos de caracteres de doble byte. Debido a que cada carácter requiere 2 bytes, la entrada, visualización e impresión de caracteres DBCS requiere hardware y software de soporte que sea compatible con DBCS.

DWARF
DWARF ha sido desarrollado por UNIX International Programming Languages Special Interest Group (SIG). Está diseñado para satisfacer las necesidades simbólicas de depuración a nivel de fuente de diferentes lenguajes de forma unificada proporcionando información de depuración independiente del lenguaje. Un archivo DWARF contiene datos de depuración organizados en distintos elementos. Para obtener más información, consulte *Información del programa DWARF* en la publicación *Referencia de biblioteca de extensiones DWARF/ELF*.

*** acceso dinámico**
Modalidad de acceso en la que se pueden obtener o colocar registros lógicos específicos en un archivo de almacenamiento masivo de forma no secuencial y se pueden obtener de un archivo de forma secuencial durante el ámbito de la misma sentencia OPEN .

Llamada dinámica
Una sentencia CALL *literal* en un programa que se ha compilado con la opciónde DYNAM , o una sentencia CALL *identificador* en un programa.

E

*** EBCDIC (Código de intercambio decimal ampliado codificado en binario)**
Juego de caracteres codificado basado en caracteres codificados de 8 bits.

Carácter EBCDIC
Cualquiera de los símbolos incluidos en el conjunto EBCDIC (Extended Binary-Coded-Decimal Interchange Code).

EBCDIC DBCS
Véase *EBCDIC de doble byte*.

elemento de datos editado
Elemento de datos que se ha modificado suprimiendo ceros o insertando caracteres de edición o ambos.

*** carácter de edición**
Un único carácter o una combinación fija de dos caracteres que pertenece al siguiente conjunto:

Carácter	Significado
	Espacio
0	ZERO
+	Signo más
-	Menos
PC	Crédito
BD	Débito
Z	Supresión de cero
*	Comprobar protección
$	Símbolo de moneda
,	Coma (coma decimal)
.	Punto (coma decimal)
/	Inclinado (barra inclinada)

elemento (elemento de texto)
Unidad lógica de una serie de texto, como la descripción de un único elemento de datos o verbo, precedida de un código exclusivo que identifica el tipo de elemento.

*** elemento elemental**
Elemento de datos que se describe como no subdividido lógicamente.

Sistema de archivos SFS deCICS
Véase *sistema de archivos SFS*.

unidad de codificación
Véase *unidad de codificación de caracteres*.

*** fin de PROCEDURE DIVISION**
La posición física de un programa fuente COBOL después de la cual no aparecen más procedimientos.

*** finalizar marcador de programa**
Combinación de palabras, seguida de un punto de separación, que indica el final de un programa fuente COBOL. El marcador de programa final es:

```
END PROGRAM program-name.
```

*** entrada**
Cualquier conjunto descriptivo de cláusulas consecutivas terminadas por un punto de separación y escritas en IDENTIFICATION DIVISION, ENVIRONMENT DIVISIONo DATA DIVISION de un programa COBOL.

*** cláusula de entorno**
Cláusula que aparece como parte de una entrada ENVIRONMENT DIVISION .

ENVIRONMENT DIVISION
Una de las cuatro partes de componente principales de un programa COBOL. El ENVIRONMENT DIVISION describe los sistemas en los que se compila el programa fuente y aquellos en los que

se ejecuta el programa objeto. Proporciona un enlace entre el concepto lógico de archivos y sus registros, y los aspectos físicos de los dispositivos en los que se almacenan los archivos.

nombre-entorno
Nombre, especificado por IBM, que identifica las unidades lógicas del sistema, los caracteres de control de perforación de impresora y tarjeta, los códigos de informe, los conmutadores de programa o todos ellos. Cuando un nombre de entorno está asociado con un nombre mnemotécnico en ENVIRONMENT DIVISION, el nombre mnemotécnico se puede sustituir en cualquier formato en el que dicha sustitución sea válida.

entorno, variable
Cualquiera de una serie de variables que definen algún aspecto del entorno informático y son accesibles a los programas que operan en dicho entorno. Las variables de entorno pueden afectar al comportamiento de los programas que son sensibles al entorno en el que operan.

tiempo de ejecución
Véase *tiempo de ejecución*.

entorno de ejecución
Véase *entorno de ejecución*.

campo de fecha expandida
Campo de fecha que contiene un año expandido (cuatro dígitos). Véase también *campo de fecha* y *año expandido*.

año expandido
Campo de fecha que consta sólo de un año de cuatro dígitos. Su valor incluye el siglo: por ejemplo, 1998. Compárese con *año de ventana*.

*** terminador de ámbito explícito**
Palabra reservada que termina el ámbito de una sentencia PROCEDURE DIVISION determinada.

exponente
Número que indica la potencia a la que se va a elevar otro número (la base). Los exponentes positivos denotan multiplicación; los exponentes negativos denotan división; y los exponentes fraccionados denotan una raíz de una cantidad. En COBOL, una expresión exponencial se indica con el símbolo ** seguido del exponente.

*** expresión**
Una expresión aritmética o condicional.

*** modalidad de ampliación**
El estado de un archivo después de la ejecución de una sentencia OPEN , con la frase EXTEND especificada para dicho archivo, y antes de la ejecución de una sentencia CLOSE , sin la frase REEL o UNIT para dicho archivo.

Lenguaje de códigos ampliable
Véase *XML*.

ampliaciones
Sintaxis y semántica COBOL soportadas por los compiladores de IBM además de las descritas en 85 COBOL Estándar.

página de códigos externa
Para documentos XML ASCII o UTF-8 , la página de códigos indicada por el entorno local de tiempo de ejecución actual. Para documentos XML EBCDIC, o bien:

- La página de códigos especificada en la variable de entorno EBCDIC_CODEPAGE
- La página de códigos EBCDIC predeterminada seleccionada para el entorno local de tiempo de ejecución actual si la variable de entorno EBCDIC_CODEPAGE no está establecida

*** datos externos**
Los datos que se describen en un programa como elementos de datos externos y conectores de archivos externos.

*** elemento de datos externo**
Elemento de datos que se describe como parte de un registro externo en uno o más programas de una unidad de ejecución y al que se puede hacer referencia desde cualquier programa en el que se describe.

*** registro de datos externo**
Registro lógico que se describe en uno o más programas de una unidad de ejecución y cuyos elementos de datos constitutivos pueden referenciarse desde cualquier programa en el que se describen.

elemento de datos decimales externos
Véase *elemento de datos decimal con zona* y *elemento de datos decimal nacional*.

*** conector de archivo externo**
Conector de archivo al que pueden acceder uno o varios programas de objeto de la unidad de ejecución.

elemento de datos de coma flotante externo
Véase *elemento de datos de coma flotante de visualización* y *elemento de datos de coma flotante nacional*.

Programa externo
El programa más externo. Un programa que no está anidado.

*** conmutador externo**
Dispositivo de hardware o software, definido y nombrado por el implementador, que se utiliza para indicar que existe uno de los dos estados alternativos.

F

*** constante figurativa**
Valor generado por el compilador al que se hace referencia mediante el uso de determinadas palabras reservadas.

*** archivo**
Colección de registros lógicos.

*** condición de conflicto de atributo de archivo**
Se ha realizado un intento no satisfactorio de ejecutar una operación de entrada-salida en un archivo y los atributos de archivo, tal como se ha especificado para ese archivo en el programa, no coinciden con los atributos fijos para ese archivo.

*** cláusula file**
Cláusula que aparece como parte de cualquiera de las siguientes entradas de `DATA DIVISION`: entrada de descripción de archivo (entrada`FD`) y entrada de descripción de archivo de fusión de clasificación (entrada`SD`).

*** conector de archivo**
Área de almacenamiento que contiene información sobre un archivo y se utiliza como enlace entre un nombre de archivo y un archivo físico y entre un nombre de archivo y su área de registro asociada.

*** entrada de control de archivo**
Una cláusula `SELECT` y todas sus cláusulas subordinadas que declaran los atributos físicos relevantes de un archivo.

`FILE-CONTROL` párrafo
Un párrafo en el `ENVIRONMENT DIVISION` en el que se declaran los archivos de datos para una unidad de origen determinada.

*** entrada de descripción de archivo**
Una entrada en `FILE SECTION` del `DATA DIVISION` que se compone del indicador de nivel `FD`, seguido de un nombre de archivo y, a continuación, seguido de un conjunto de cláusulas de archivo según sea necesario.

*** nombre-archivo**
Palabra definida por el usuario que nombra un conector de archivo descrito en una entrada de descripción de archivo o una entrada de descripción de archivo de fusión de clasificación dentro del `FILE SECTION` de `DATA DIVISION`.

*** organización de archivos**
Estructura de archivo lógico permanente establecida en el momento en que se crea un archivo.

indicador de posición de archivo
Entidad conceptual que contiene el valor de la clave actual dentro de la clave de referencia para un archivo indexado, o el número de registro del registro actual para un archivo secuencial, o el número de registro relativo del registro actual para un archivo relativo, o indica que no existe ningún registro lógico siguiente, o que no está disponible un archivo de entrada opcional, o que ya existe la condición `AT END`, o que no se ha establecido ningún registro siguiente válido.

*** `FILE SECTION`**
La sección de `DATA DIVISION` que contiene entradas de descripción de archivo y entradas de descripción de archivo de fusión de clasificación junto con sus descripciones de registro asociadas.

Sistema de archivos
Colección de archivos que se ajustan a un conjunto específico de protocolos de registro de datos y de descripción de archivos, y a un conjunto de programas que gestionan estos archivos.

*** atributos de archivo fijo**
Información sobre un archivo que se establece cuando se crea un archivo y que no se puede cambiar posteriormente durante la existencia del archivo. Estos atributos incluyen la organización del archivo (secuencial, relativo o indexado), la clave de registro principal, las claves de registro alternativas, el conjunto de códigos, el tamaño mínimo y máximo de registro, el tipo de registro (fijo o variable), la secuencia de clasificación de las claves para archivos indexados, el factor de bloqueo, el carácter de relleno y el delimitador de registro.

*** registro de longitud fija**
Un registro asociado a un archivo cuya descripción de archivo o entrada de descripción de fusión de clasificación requiere que todos los registros contengan el mismo número de bytes.

elemento de punto fijo
Elemento de datos numérico definido con una cláusula `PICTURE` que especifica la ubicación de un signo opcional, el número de dígitos que contiene y la ubicación de una coma decimal opcional. El formato puede ser binario, decimal empaquetado o decimal externo.

indicadores de comentario flotante (* >)
Un indicador de comentario flotante indica una línea de comentario si es la primera serie de caracteres en el área de texto de programa (Área A más Área B), o indica un comentario en línea si está detrás de una o más series de caracteres en el área de texto de programa.

floating point
Formato para representar números en el que un número real se representa mediante un par de números distintos. En una representación de coma flotante, el número real es el producto de la parte de punto fijo (el primer numeral) y un valor obtenido elevando la base de coma flotante implícita a una potencia indicada por el exponente (el segundo numeral). Por ejemplo, una representación de coma flotante del número 0.0001234 es 0.1234 -3, donde 0.1234 es la mantisa y -3 es el exponente.

elemento de datos de coma flotante
Elemento de datos numérico que contiene una fracción y un exponente. Su valor se obtiene multiplicando la fracción por la base del elemento de datos numérico elevado a la potencia que el exponente especifica.

*** formato**
Organización específica de un conjunto de datos.

*** función**
Elemento de datos temporal cuyo valor se determina en el momento en que se hace referencia a la función durante la ejecución de una sentencia.

*** identificador-función**
Combinación sintácticamente correcta de series de caracteres y separadores que hacen referencia a una función. El elemento de datos representado por una función se identifica de forma exclusiva mediante un nombre de función con sus argumentos, si los hay. Un identificador de función puede incluir un modificador de referencia. Un identificador de función que hace referencia a una función alfanumérica se puede especificar en cualquier lugar de los formatos generales en los que se puede especificar un identificador, sujeto a determinadas restricciones. Se puede hacer referencia a un

identificador de función que hace referencia a una función entera o numérica en cualquier lugar de los formatos generales en los que se puede especificar una expresión aritmética.

nombre-función
Palabra que nombra el mecanismo cuya invocación, junto con los argumentos necesarios, determina el valor de una función.

elemento de datos de puntero de función
Elemento de datos en el que se puede almacenar un puntero a un punto de entrada. Un elemento de datos definido con la cláusula `USAGE IS FUNCTION-POINTER` contiene la dirección de un punto de entrada de función. Normalmente se utiliza para comunicarse con programas C y Java .

G

Recogida de basura
La liberación automática por parte del sistema de tiempo de ejecución Java de la memoria para objetos a los que ya no se hace referencia.

GDG
Véase *grupo de datos de generación (GDG)*.

GDS
Véase *conjunto de datos de generación (GDS)*.

grupo de datos de generación (GDG)
Colección de archivos relacionados cronológicamente; cada uno de estos archivos se denomina un *conjunto de datos de generación (GDS)* o generación.

conjunto de datos de generación (GDS)
Uno de los archivos de un *grupo de datos de generación (GDG)*; cada archivo de este tipo está relacionado cronológicamente con los otros archivos del grupo.

*** nombre global**
Nombre que se declara en un solo programa pero al que se puede hacer referencia desde el programa y desde cualquier programa contenido en el programa. Los nombres de condición, los nombres de datos, los nombres de archivo, los nombres de registro, los nombres de informe y algunos registros especiales pueden ser nombres globales.

elemento de grupo
(1) Un elemento de datos que se compone de elementos de datos subordinados. Véase *elemento de grupo alfanumérico* y *elemento de grupo nacional*. (2) Cuando no se califique explícitamente o por contexto como grupo nacional o grupo alfanumérico, el término se refiere a los grupos en general.

separador de agrupación
Carácter utilizado para separar unidades de dígitos en números para facilitar la lectura. El valor predeterminado es la coma de carácter.

H

etiqueta de cabecera
(1) Una etiqueta de que precede a los registros de datos en una unidad de soporte de grabación. (2) Sinónimo de *etiqueta de inicio de archivo*.

*** fin de orden superior**
El carácter situado más a la izquierda de una serie de caracteres.

elemento de datos alfanuméricos de host
(De documentos XML) Elemento de datos alfanuméricos de categoría cuya entrada de descripción de datos no contiene la frase `NATIVE` y que se ha compilado con la opción `CHAR(EBCDIC)` en vigor. La codificación del elemento de datos es la página de códigos EBCDIC en vigor. Esta página de códigos se determina a partir de la variable de entorno EBCDIC_CODEPAGE, si se establece, de lo contrario a partir de la página de códigos predeterminada asociada con el entorno local de ejecución.

I

IBM Extensión COBOL
Sintaxis y semántica COBOL soportadas por los compiladores de IBM además de las descritas en 85 COBOL Estándar.

ICU
Consulte *International Components for Unicode (ICU)*.

IDENTIFICATION DIVISION
Una de las cuatro partes de componente principales de un programa COBOL. `IDENTIFICATION DIVISION` identifica el programa, la clase. `IDENTIFICATION DIVISION` puede incluir la siguiente documentación: nombre de autor, instalación o fecha.

*** identificador**
Combinación sintácticamente correcta de series de caracteres y separadores que nombra un elemento de datos. Al hacer referencia a un elemento de datos que no es una función, un identificador consta de un nombre de datos, junto con sus calificadores, subíndices y modificador de referencia, según sea necesario para la exclusividad de la referencia. Cuando se hace referencia a un elemento de datos que es una función, se utiliza un identificador de función.

*** sentencia imperativa**
Sentencia que empieza con un verbo imperativo y especifica una acción incondicional que se debe realizar o es una sentencia condicional delimitada por su terminador de ámbito explícito (sentencia de ámbito delimitado). Una sentencia imperativa puede consistir en una secuencia de sentencias imperativas.

*** terminador de ámbito implícito**
Un punto de separación que termina el ámbito de cualquier sentencia no terminada anterior, o una frase de una sentencia que por su aparición indica el final del ámbito de cualquier sentencia contenida en la frase anterior.

*** índice**
Un área de almacenamiento de computadora o registro, cuyo contenido representa la identificación de un elemento particular en una tabla.

*** elemento de datos de índice**
Elemento de datos en el que los valores asociados a un nombre de índice se pueden almacenar en un formato especificado por el implementador.

nombre-datos-indexado
Identificador compuesto de un nombre de datos, seguido de uno o más nombres de índice entre paréntesis.

*** archivo indexado**
Un archivo con una organización indexada.

*** organización indexada**
Estructura de archivo lógico permanente en la que cada registro se identifica mediante el valor de una o más claves dentro de ese registro.

indexación
Sinónimo de *subscripción* utilizando nombres de índice.

*** nombre-índice**
Palabra definida por el usuario que nombra un índice asociado a una tabla específica.

*** programa inicial**
Programa que se coloca en un estado inicial cada vez que se llama al programa en una unidad de ejecución.

*** estado inicial**
El estado de un programa cuando se llama por primera vez en una unidad de ejecución.

inline
En un programa, instrucciones que se ejecutan secuencialmente, sin ramificar a rutinas, subrutinas u otros programas.

*** archivo de entrada**
Un archivo que se abre en la modalidad de entrada.

*** modalidad de entrada**
El estado de un archivo después de la ejecución de una sentencia OPEN , con la frase INPUT especificada, para dicho archivo y antes de la ejecución de una sentencia CLOSE , sin la frase REEL o UNIT para dicho archivo.

*** archivo de entrada-salida**
Un archivo que se abre en modalidad I-O .

*** INPUT-OUTPUT SECTION**
La sección de ENVIRONMENT DIVISION que nombra los archivos y el soporte externo que necesita un programa objeto y que proporciona información necesaria para la transmisión y el manejo de datos en tiempo de ejecución.

*** sentencia de entrada-salida**
Sentencia que hace que los archivos se procesen realizando operaciones en registros individuales o en el archivo como una unidad. Las sentencias de entrada-salida son ACCEPT (con la frase de identificador), CLOSE, DELETE, DISPLAY, OPEN, READ, REWRITE, SET (con la frase TO ON o TO OFF), STARTy WRITE.

*** procedimiento de entrada**
Un conjunto de sentencias, al que se proporciona control durante la ejecución de una sentencia SORT , con el fin de controlar la liberación de los registros especificados que se van a ordenar.

*** entero**
(1) Un literal numérico que no incluye ninguna posición de dígito a la derecha de la coma decimal. (2) Un elemento de datos numérico definido en DATA DIVISION que no incluye ninguna posición de dígito a la derecha de la coma decimal. (3) Una función numérica cuya definición establece que todos los dígitos a la derecha de la coma decimal son cero en el valor devuelto para cualquier posible evaluación de la función.

función de entero
Función cuya categoría es numérica y cuya definición no incluye ninguna posición de dígito a la derecha de la coma decimal.

comunicación entre lenguajes (ILC)
La capacidad de las rutinas escritas en diferentes lenguajes de programación para comunicarse. El soporte de ILC le permite crear fácilmente aplicaciones a partir de rutinas de componentes escritas en diversos idiomas.

resultado intermedio
Campo intermedio que contiene los resultados de una sucesión de operaciones aritméticas.

*** datos internos**
Los datos que se describen en un programa y excluyen todos los elementos de datos externos y conectores de archivos externos. Los elementos descritos en LINKAGE SECTION de un programa se tratan como datos internos.

*** elemento de datos interno**
Elemento de datos que se describe en un programa en una unidad de ejecución. Un elemento de datos interno puede tener un nombre global.

elemento de datos decimal interno
Elemento de datos que se describe como USAGE PACKED-DECIMAL o USAGE COMP-3y que tiene una serie de caracteres PICTURE que define el elemento como numérico (una combinación válida de símbolos 9, S, Po V). Sinónimo de *elemento de datos decimal empaquetado*.

*** conector de archivo interno**
Un conector de archivo al que sólo puede acceder un programa objeto de la unidad de ejecución.

elemento de datos de coma flotante interno
Elemento de datos que se describe como USAGE COMP-1 o USAGE COMP-2. COMP-1 define un elemento de datos de coma flotante de precisión simple. COMP-2 define un elemento de datos de coma flotante de precisión doble. No hay ninguna cláusula PICTURE asociada con un elemento de datos de coma flotante interno.

Componentes internacionales para Unicode (ICU)
Un proyecto de desarrollo de código abierto patrocinado, soportado y utilizado por IBM. Las bibliotecas ICU proporcionan servicios Unicode robustos y con todas las características en una amplia variedad de plataformas, incluyendo AIX y Linux.

*** estructura de datos intraregistro**
Colección completa de grupos y elementos de datos elementales de un registro lógico que define un subconjunto contiguo de las entradas de descripción de datos. Estas entradas de descripción de datos incluyen todas las entradas cuyo número de nivel es mayor que el número de nivel de la primera entrada de descripción de datos que describe la estructura de datos dentro del registro.

función intrínseca
Una función predefinida, como una función aritmética de uso común, llamada por una referencia de función incorporada.

*** condición de clave no válida**
Una condición, en tiempo de ejecución, se produce cuando se determina que un valor específico de la clave asociada con un archivo indexado o relativo no es válido.

*** I-O-CONTROL**
El nombre de un párrafo ENVIRONMENT DIVISION en el que se especifican los requisitos de programa de objeto para volver a ejecutar puntos, el uso compartido de las mismas áreas por varios archivos de datos y el almacenamiento de varios archivos en un único dispositivo de entrada-salida.

*** I-O-CONTROL entrada**
Una entrada en el párrafo I-O-CONTROL de ENVIRONMENT DIVISION; esta entrada contiene cláusulas que proporcionan la información necesaria para la transmisión y el manejo de datos en archivos con nombre durante la ejecución de un programa.

*** Modalidad I-O**
El estado de un archivo después de la ejecución de una sentencia OPEN , con la frase I-O especificada, para dicho archivo y antes de la ejecución de una sentencia CLOSE sin la fase REEL o UNIT para dicho archivo.

*** Estado de I-O**
Entidad conceptual que contiene el valor de dos caracteres que indica el estado resultante de una operación de entrada-salida. Este valor se pone a disposición del programa mediante el uso de la cláusula FILE STATUS en la entrada de control de archivos para el archivo.

estructura de iteración
Lógica de proceso de programa en la que se repite una serie de sentencias mientras una condición es verdadera o hasta que una condición es verdadera.

J

J2EE
Consulte *Java 2 Platform, Enterprise Edition (J2EE)*.

Java 2 Platform, Enterprise Edition (J2EE)
Entorno para el desarrollo y el despliegue de aplicaciones de empresa, definido por Oracle. La plataforma J2EE consta de un conjunto de servicios, interfaces de programación de aplicaciones (API) y protocolos que proporcionan la funcionalidad para desarrollar aplicaciones basadas en web de varios niveles. (Oracle)

Java Native Interface (JNI)
Interfaz de programación que permite que el código Java que se ejecuta dentro de una máquina virtual Java (JVM) interopere con aplicaciones y bibliotecas escritas en otros lenguajes de programación.

Máquina virtualJava (JVM)
Implementación de software de una unidad de proceso central que ejecuta programas Java compilados.

JSON
JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos.

JVM
Consulte *Máquina virtualJava (JVM)*.

K

K
Al hacer referencia a la capacidad de almacenamiento, dos a la décima potencia; 1024 en notación decimal.

*** clave**
Elemento de datos que identifica la ubicación de un registro o un conjunto de elementos de datos que sirven para identificar el orden de los datos.

*** clave de referencia**
La clave, ya sea primaria o alternativa, que se está utilizando actualmente para acceder a los registros dentro de un archivo indexado.

*** palabra clave**
Palabra sensible al contexto o palabra reservada cuya presencia es necesaria cuando el formato en el que aparece la palabra se utiliza en una unidad de origen.

kilobyte (KB)
Un kilobyte equivale a 1024 bytes.

L

*** nombre-idioma**
Nombre de sistema que especifica un lenguaje de programación determinado.

último estado utilizado
Un estado en el que se encuentra un programa si sus valores internos siguen siendo los mismos que cuando se salió del programa (los valores no se restablecen a sus valores iniciales).

*** letra**
Un carácter que pertenece a uno de los dos conjuntos siguientes:

1. Letras mayúsculas: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
2. Letras minúsculas: a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z

*** indicador de nivel**
Dos caracteres alfabéticos que identifican un tipo específico de archivo o una posición en una jerarquía. Los indicadores de nivel en `DATA DIVISION` son: `CD`, `FD`y `SD`.

*** número-nivel**
Palabra definida por el usuario (expresada como un número de dos dígitos) que indica la posición jerárquica de un elemento de datos o las propiedades especiales de una entrada de descripción de datos. Los números de nivel en el rango de 1 a 49 indican la posición de un elemento de datos en la estructura jerárquica de un registro lógico. Los números de nivel en el rango de 1 a 9 se pueden escribir como un solo dígito o como un cero seguido de un dígito significativo. Los números de nivel 66, 77 y 88 identifican propiedades especiales de una entrada de descripción de datos.

*** nombre-biblioteca**
Palabra definida por el usuario que nombra una biblioteca COBOL que el compilador va a utilizar para compilar un programa fuente determinado.

*** texto de biblioteca**
Una secuencia de palabras de texto, líneas de comentario, el espacio de separador o el delimitador de pseudotexto de separador en una biblioteca COBOL.

Fecha de Lilian
El número de días desde el inicio del calendario gregoriano. El primer día es el viernes 15 de octubre de 1582. El formato de fecha Lilian es nombrado en honor a Luigi Lilio, el creador del calendario gregoriano.

*** contador de linaje**
Un registro especial cuyo valor apunta a la posición actual dentro del cuerpo de la página.

link
(1) La combinación de la conexión de enlace (el medio de transmisión) y dos estaciones de enlace, una en cada extremo de la conexión de enlace. Un enlace se puede compartir entre varios enlaces en una configuración de multipunto o red en anillo. (2) Interconectar elementos de datos o partes de uno o varios programas de sistema; por ejemplo, enlazar programas de objeto mediante un editor de enlaces para generar una biblioteca compartida.

LINKAGE SECTION
La sección del DATA DIVISION del programa llamado que describe los elementos de datos disponibles en el programa de llamada. Tanto el programa de llamada como el programa llamado pueden hacer referencia a estos elementos de datos.

literal
Serie de caracteres cuyo valor se especifica mediante el conjunto ordenado de caracteres que comprende la serie o mediante el uso de una constante figurativa.

little-endian
El formato predeterminado que utilizan los procesadores Intel para almacenar datos binarios y caracteres UTF-16 . En este formato, el byte más significativo de un elemento de datos binarios está en la dirección más alta y el byte más significativo de un carácter UTF-16 está en la dirección más alta. Compárese con *big-endian*.

Entorno local
Conjunto de atributos para un entorno de ejecución de programa que indica consideraciones culturalmente sensibles, como la página de códigos de caracteres, la secuencia de clasificación, el formato de fecha y hora, la representación de valores monetarios, la representación de valores numéricos o el idioma.

*** LOCAL-STORAGE SECTION**
La sección del DATA DIVISION que define el almacenamiento que se asigna y libera por invocación, en función del valor asignado en las cláusulas VALUE .

*** operador lógico**
Una de las palabras reservadas AND, OR o NOT. En la formación de una condición, ya sea AND, o OR, o ambos se pueden utilizar como conectivos lógicos. NO se puede utilizar para la negación lógica.

*** registro lógico**
El elemento de datos más inclusivo. El número de nivel de un registro es 01. Un registro puede ser un elemento elemental o un grupo de elementos. Sinónimo de *registro*.

*** fin de orden inferior**
El carácter situado más a la derecha de una serie de caracteres.

Sistema de archivos LSQ
El sistema de archivos LSQ sólo admite archivos LINE SEQUENTIAL.

m

Programa principal
En una jerarquía de programas y subrutinas, el primer programa que recibe el control cuando los programas se ejecutan dentro de un proceso.

makefile
Archivo de texto que contiene una lista de los archivos de la aplicación. El programa de utilidad make utiliza este archivo para actualizar los archivos de destino con los últimos cambios.

*** almacenamiento masivo**
Medio de almacenamiento en el que los datos se pueden organizar y mantener de forma secuencial y no secuencial.

*** dispositivo de almacenamiento masivo**
Dispositivo que tiene una gran capacidad de almacenamiento, como un disco magnético.

*** archivo de almacenamiento masivo**
Colección de registros que se almacena en un medio de almacenamiento masivo.

MBCS
Véase *juego de caracteres multibyte (MBCS)*.

*** megabyte (MB)**
Un megabyte equivale a 1.048.576 bytes.

*** archivo de fusión**
Colección de registros que una sentencia MERGE debe fusionar. El archivo de fusión se crea y sólo puede ser utilizado por la función de fusión.

*** nombre-mnemotécnico**
Palabra definida por el usuario que está asociada en ENVIRONMENT DIVISION con un nombre de implementación especificado.

archivo de definición de módulo
Archivo que describe los segmentos de código dentro de un módulo de carga.

carácter multibyte
Cualquier carácter que esté representado en 2 o más bytes en un juego de caracteres de varios bytes. Por ejemplo, un carácter DBCS o cualquier carácter UTF-8 que esté representado en dos o más bytes. Los caracteres UTF-16 no son caracteres multibyte porque UTF-16 no es un juego de caracteres multibyte.

juego de caracteres multibyte (MBCS)
Juego de caracteres codificado que se compone de caracteres representados en un número variable de bytes. Los ejemplos son: EUC (Extended Unix Code), UTF-8y juegos de caracteres compuestos de una mezcla de caracteres EBCDIC o ASCII de un solo byte y de doble byte.

multitarea
Modalidad de operación que proporciona la ejecución simultánea o intercalada de dos o más tareas.

multihebra
Operación simultánea de más de una vía de acceso de ejecución dentro de un sistema. Sinónimo de *multiproceso*.

N

nombre
Palabra (compuesta de no más de 30 caracteres) que define un operando COBOL.

espacio de nombres
Véase *espacio de nombres XML*.

carácter nacional
(1) Un carácter UTF-16 en un elemento de datos USAGE NATIONAL o literal nacional. (2) Cualquier carácter representado en UTF-16.

datos de caracteres nacionales
Referencia general a los datos representados en UTF-16.

posición de carácter nacional
Véase *posición de carácter*.

datos nacionales
Véase *datos de caracteres nacionales*.

elemento de datos nacional
Un elemento de datos de categoría nacional, editado a nivel nacional o editado a nivel numérico de USAGE NATIONAL.

elemento de datos decimales nacionales
Elemento de datos decimal externo que se describe implícita o explícitamente como USAGE NATIONAL y que contiene una combinación válida de PICTURE símbolos 9, S, Py V.

elemento de datos editado a nivel nacional
Elemento de datos que se describe mediante una serie de caracteres PICTURE que contiene al menos una instancia del símbolo N y al menos uno de los símbolos de inserción simples B, 0o /. Un elemento de datos editado a nivel nacional tiene USAGE NATIONAL.

elemento de datos de coma flotante nacional
Elemento de datos de coma flotante externo que se describe implícita o explícitamente como USAGE NATIONAL y que tiene una serie de caracteres PICTURE que describe un elemento de datos de coma flotante.

elemento de grupo nacional
Elemento de grupo que se describe explícita o implícitamente con una cláusula GROUP-USAGE NATIONAL . Un elemento de grupo nacional se procesa como si se hubiera definido como un elemento de datos elemental de categoría nacional para operaciones como INSPECT, STRINGy UNSTRING. Este proceso garantiza el relleno y truncamiento correctos de los caracteres nacionales, en contraste con la definición de elementos de datos de USAGE NATIONAL dentro de un elemento de grupo alfanumérico. Para las operaciones que requieren el proceso de los elementos elementales dentro de un grupo, como MOVE CORRESPONDING, ADD CORRESPONDINGy INITIALIZE, un grupo nacional se procesa utilizando la semántica de grupo.

elemento de datos alfanuméricos nativos
(De documentos XML) Elemento de datos alfanuméricos de categoría que se describe con la frase NATIVE o que se ha compilado con la opción CHAR(NATIVE) en vigor. La codificación del elemento de datos es la página de códigos ASCII o UTF-8 del entorno local de ejecución en vigor.

*** juego de caracteres nativo**
El juego de caracteres definido por el implementador asociado con el sistema especificado en el párrafo OBJECT-COMPUTER .

*** secuencia de clasificación nativa**
El orden de clasificación definido por el implementador asociado con el sistema especificado en el párrafo OBJECT-COMPUTER .

*** condición combinada negada**
El operador lógico NOT seguido inmediatamente por una condición combinada entre paréntesis. Véase también *condición* y *condición combinada*.

*** condición simple negada**
El operador lógico NOT seguido inmediatamente por una condición simple. Véase también *condición* y *condición simple*.

programa anidado
Programa que está directamente contenido en otro programa.

*** siguiente frase ejecutable**
Se completa la siguiente frase a la que se transferirá el control después de la ejecución de la sentencia actual.

*** siguiente sentencia ejecutable**
La siguiente sentencia a la que se transferirá el control después de que se haya completado la ejecución de la sentencia actual.

*** registro siguiente**
El registro que sigue lógicamente al registro actual de un archivo.

*** elementos no contiguos**
Elementos de datos elementales en WORKING-STORAGE SECTION y LINKAGE SECTION que no tienen ninguna relación jerárquica con otros elementos de datos.

no-fecha
Cualquiera de los elementos siguientes:

- Un elemento de datos cuya entrada de descripción de fecha no incluye la cláusula DATE FORMAT
- Un literal
- Un campo de fecha que se ha convertido utilizando la función UNDATE
- Un campo de fecha de modificación de referencia
- El resultado de determinadas operaciones aritméticas que pueden incluir operandos de campo de fecha; por ejemplo, la diferencia entre dos campos de fecha compatibles

NULL
Constante figurativa que se utiliza para asignar, a elementos de datos de puntero, el valor de una dirección que no es válida. NULLS se puede utilizar siempre que se pueda utilizar NULL .

*** carácter numérico**
Carácter que pertenece al siguiente conjunto de dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

elemento de datos numéricos
(1) Un elemento de datos cuya descripción restringe su contenido a un valor representado por caracteres elegidos entre los dígitos del 0 al 9. Si está firmado, el elemento también puede contener un signo +,-u otra representación de un signo operativo. (2) Un elemento de datos de categoría numérica, de coma flotante interna o de coma flotante externa. Un elemento de datos numérico puede tener USAGE DISPLAY, NATIONAL, PACKED-DECIMAL, BINARY, COMP, COMP-1, COMP-2, COMP-3, COMP-4o COMP-5.

elemento de datos editado numérica-editado
Elemento de datos que contiene datos numéricos en un formato adecuado para su uso en la salida impresa. El elemento de datos puede constar de dígitos decimales externos de 0 a 9, el separador decimal, comas, el signo de moneda, caracteres de control de signo y otros caracteres de edición. Un elemento editado numérica-se puede representar en USAGE DISPLAY o USAGE NATIONAL.

*** función numérica**
Una función cuya clase y categoría son numéricas pero que para alguna posible evaluación no satisface los requisitos de las funciones enteras.

*** literal numérico**
Literal compuesto de uno o más caracteres numéricos que pueden contener una coma decimal o un signo algebraico, o ambos. La coma decimal no debe ser el carácter situado más a la derecha. El signo algebraico, si está presente, debe ser el carácter más a la izquierda.

O

código de objeto
Salida de un compilador o ensamblador que es en sí mismo código de máquina ejecutable o es adecuado para el proceso para producir código de máquina ejecutable.

*** OBJECT-COMPUTER**
El nombre de un párrafo ENVIRONMENT DIVISION en el que se describe el entorno del sistema, donde se ejecuta el programa objeto.

*** entrada de sistema de objetos**
Una entrada en el párrafo OBJECT-COMPUTER de ENVIRONMENT DIVISION; esta entrada contiene cláusulas que describen el entorno del sistema en el que se va a ejecutar el programa objeto.

*** objeto de entrada**
Conjunto de operandos y palabras reservadas, dentro de una entrada DATA DIVISION de un programa COBOL, que sigue inmediatamente al asunto de la entrada.

programa objeto
Conjunto o grupo de instrucciones de lenguaje de máquina ejecutables y otro material diseñado para interactuar con los datos para proporcionar soluciones de problemas. En este contexto, un programa objeto es generalmente el resultado de lenguaje de máquina de la operación de un compilador COBOL en una definición de programa fuente . Cuando no hay peligro de ambigüedad, se puede utilizar la palabra *programa* en lugar de *programa objeto*.

*** hora de objeto**
La hora a la que se ejecuta un programa objeto. Sinónimo de *tiempo de ejecución*.

*** elemento obsoleto**
Un elemento de lenguaje COBOL en 85 COBOL Estándar que se ha suprimido de Estándar COBOL 2002.

ODBC
Véase *Open Database Connectivity (ODBC)*.

Objeto ODO
En el ejemplo siguiente, X es el objeto de la cláusula OCCURS DEPENDING ON (objeto ODO).

```
WORKING-STORAGE SECTION.
01  TABLE-1.
    05  X                   PIC S9.
    05  Y OCCURS 3 TIMES
          DEPENDING ON X    PIC X.
```

El valor del objeto ODO determina cuántos de los sujetos ODO aparecen en la tabla.

Asunto de ODO
En el ejemplo anterior, Y es el sujeto de la cláusula OCCURS DEPENDING ON (sujeto ODO). El número de sujetos ODO de Y que aparecen en la tabla depende del valor de X.

Conectividad de base de datos abierta (ODBC)
Especificación para una interfaz de programación de aplicaciones (API) que proporciona acceso a datos en una variedad de bases de datos y sistemas de archivos.

*** modalidad abierta**
El estado de un archivo después de la ejecución de una sentencia OPEN para dicho archivo y antes de la ejecución de una sentencia CLOSE sin la frase REEL o UNIT para dicho archivo. La modalidad de apertura concreta se especifica en la sentencia OPEN como INPUT, OUTPUT, I-Oo EXTEND.

*** operando**
(1) La definición general del operando es "el componente sobre el que se opera." (2) A los efectos del presente documento, cualquier palabra (o palabras) en minúscula que aparezca en una declaración o formato de entrada puede considerarse un operando y, como tal, es una referencia implícita a los datos indicados por el operando.

Operación
Servicio que se puede solicitar de un objeto.

*** signo operativo**
Signo algebraico asociado con un elemento de datos numérico o un literal numérico, para indicar si su valor es positivo o negativo.

archivo opcional
Un archivo que se declara como no necesariamente disponible cada vez que se ejecuta el programa objeto.

*** palabra opcional**
Palabra reservada que se incluye en un formato específico sólo para mejorar la legibilidad del idioma. Su presencia es opcional para el usuario cuando el formato en el que aparece la palabra se utiliza en una unidad de origen.

*** archivo de salida**
Archivo que se abre en modalidad de salida o en modalidad de ampliación.

*** modalidad de salida**
El estado de un archivo después de la ejecución de una sentencia OPEN , con la frase OUTPUT o EXTEND especificada, para dicho archivo y antes de la ejecución de una sentencia CLOSE sin la frase REEL o UNIT para dicho archivo.

*** procedimiento de salida**
Conjunto de sentencias a las que se otorga control durante la ejecución de una sentencia SORT después de que se haya completado la función de clasificación, o durante la ejecución de una sentencia MERGE después de que la función de fusión alcance un punto en el que pueda seleccionar el siguiente registro en orden fusionado cuando se le solicite.

condición de desbordamiento
Condición que se produce cuando una parte del resultado de una operación excede la capacidad de la unidad de almacenamiento prevista.

P

elemento de datos decimal empaquetado
Véase *elemento de datos decimal interno*.

carácter de relleno
Carácter alfanumérico o nacional que se utiliza para rellenar las posiciones de caracteres no utilizadas en un registro físico.

página
División vertical de los datos de salida que representa una separación física de los datos. La separación se basa en requisitos lógicos internos o características externas del medio de salida o ambos.

*** cuerpo de página**
Parte de la página lógica en la que se pueden escribir o espaciar líneas o ambas.

*** párrafo**
En PROCEDURE DIVISION, un nombre de párrafo seguido de un punto de separación y de cero, una o más frases. En IDENTIFICATION DIVISION y ENVIRONMENT DIVISION, una cabecera de párrafo seguida de cero, una o más entradas.

*** cabecera de párrafo**
Palabra reservada, seguida del punto separador, que indica el principio de un párrafo en IDENTIFICATION DIVISION y ENVIRONMENT DIVISION. Las cabeceras de párrafo permitidas en IDENTIFICATION DIVISION son:

```
PROGRAM-ID. (Program IDENTIFICATION
   DIVISION)
AUTHOR.
INSTALLATION.
DATE-WRITTEN.
DATE-COMPILED.
SECURITY.
```

Las cabeceras de párrafo permitidas en ENVIRONMENT DIVISION son:

```
SOURCE-COMPUTER.
OBJECT-COMPUTER.
SPECIAL-NAMES.
REPOSITORY. (Program
   CONFIGURATION SECTION)
FILE-CONTROL.
I-O-CONTROL.
```

*** nombre-párrafo**
Palabra definida por el usuario que identifica e inicia un párrafo en PROCEDURE DIVISION.

Parámetro
Datos pasados entre un programa de llamada y un programa llamado.

*** frase**
Conjunto ordenado de una o más series de caracteres COBOL consecutivas que forman una parte de una sentencia de procedimiento COBOL o de una cláusula COBOL.

*** registro físico**
Véase *bloque*.

elemento de datos de puntero
Elemento de datos en el que se pueden almacenar los valores de dirección. Los elementos de datos se definen explícitamente como punteros con la cláusula USAGE IS POINTER . Los registros especiales de ADDRESS OF se definen implícitamente como elementos de datos de puntero. Los elementos de datos de puntero pueden compararse por igualdad o moverse a otros elementos de datos de puntero.

port
(1) Modificar un programa informático para que pueda ejecutarse en una plataforma diferente. (2) En el conjunto de protocolos de Internet, un conector lógico específico entre el Protocolo de Control de Transmisión (TCP) o el Protocolo de Datagramas de Usuario (UDP) y un protocolo o aplicación de nivel superior. Un puerto se identifica mediante un número de puerto.

Portabilidad
La capacidad de transferir un programa de aplicación de una plataforma de aplicación a otra con relativamente pocos cambios en el programa fuente.

*** clave de registro principal**
Clave cuyo contenido identifica de forma exclusiva un registro dentro de un archivo indexado.

*** número-prioridad**
Palabra definida por el usuario que clasifica secciones en PROCEDURE DIVISION para fines de segmentación. Los números de segmento sólo pueden contener los caracteres del 0 al 9. Un número de segmento puede expresarse como uno o dos dígitos.

*** procedimiento**
Un párrafo o grupo de párrafos lógicamente sucesivos, o una sección o grupo de secciones lógicamente sucesivos, dentro de PROCEDURE DIVISION.

*** sentencia de ramificación de procedimiento**
Sentencia que provoca la transferencia explícita del control a una sentencia distinta de la siguiente sentencia ejecutable en la secuencia en la que se escriben las sentencias en el código fuente. Las sentencias de ramificación de procedimiento son: ALTER, CALL, EXIT, EXIT PROGRAM, GO TO, MERGE (con la frase OUTPUT PROCEDURE), PERFORM y SORT (con la frase INPUT PROCEDURE o OUTPUT PROCEDURE), XML PARSE.

PROCEDURE DIVISION
La división COBOL que contiene instrucciones para resolver un problema.

integración de procedimientos
Una de las funciones del optimizador COBOL es simplificar las llamadas a procedimientos realizados o programas contenidos.

La integración de procedimientos PERFORM es el proceso por el que una sentencia PERFORM se sustituye por sus procedimientos realizados. La integración de procedimiento de programa contenido es el proceso en el que se sustituye una llamada a un programa contenido por el código de programa.

*** nombre-procedimiento**
Palabra definida por el usuario que se utiliza para nombrar un párrafo o sección en PROCEDURE DIVISION. Consta de un nombre de párrafo (que puede calificarse) o un nombre de sección.

Puntero de procedimiento
Elemento de datos en el que se puede almacenar un puntero a un punto de entrada. Un elemento de datos definido con la cláusula USAGE IS PROCEDURE-POINTER contiene la dirección de un punto de entrada de procedimiento.

elemento de datos de puntero de procedimiento
Elemento de datos en el que se puede almacenar un puntero a un punto de entrada. Un elemento de datos definido con la cláusula USAGE IS PROCEDURE-POINTER contiene la dirección de un punto de entrada de procedimiento. Normalmente se utiliza para comunicarse con programas COBOL .

proceso
El curso de los sucesos que se producen durante la ejecución de todo o parte de un programa. Varios procesos se pueden ejecutar simultáneamente y los programas que se ejecutan dentro de un proceso pueden compartir recursos.

programa
(1) Una secuencia de instrucciones adecuadas para ser procesadas por un ordenador. El proceso puede incluir el uso de un compilador para preparar el programa para su ejecución, así como un entorno de ejecución para ejecutarlo. (2) Conjunto lógico de uno o más módulos interrelacionados. Se pueden ejecutar varias copias del mismo programa en distintos procesos.

*** entrada de identificación de programa**
En el párrafo PROGRAM-ID de IDENTIFICATION DIVISION, entrada que contiene cláusulas que especifican el nombre de programa y asignan atributos de programa seleccionados al programa.

nombre-programa
En el IDENTIFICATION DIVISION y el marcador de programa final, una palabra definida por el usuario o un literal alfanumérico que identifica un programa fuente COBOL.

Proyecto
Conjunto completo de datos y acciones necesarios para crear un destino, como por ejemplo una biblioteca de enlaces dinámicos (DLL) u otro ejecutable (EXE).

*** pseudo-texto**
Una secuencia de palabras de texto, líneas de comentario, o el espacio de separador en un programa fuente o biblioteca COBOL limitada por, pero sin incluir, delimitadores de pseudotexto.

*** delimitador de pseudotexto**
Dos caracteres de signo igual contiguos (==) utilizados para delimitar el pseudo-texto.

*** carácter de puntuación**
Un carácter que pertenece al conjunto siguiente:

Carácter	Significado
,	Coma
;	Punto y coma
:	Dos puntos
.	Periodo (parada completa)
"	comilla
(	Paréntesis izquierdo
)	Paréntesis derecho
	Espacio
=	Signo igual

Q

QSAM (Método de acceso secuencial en cola)
Versión ampliada del método de acceso secuencial básico (BSAM). Cuando se utiliza este método, se forma una cola de bloques de datos de entrada que están a la espera de proceso o de bloques de datos de salida que se han procesado y que están a la espera de transferencia al almacenamiento auxiliar o a un dispositivo de salida.

Sistema de archivos QSAM
El sistema de archivos QSAM (método de acceso secuencial en cola) da soporte a registros fijos, variables y expandidos, y le permite acceder directamente a un archivo QSAM que ha transferido (utilizando z/OS FTP) de z/OS a AIX o Linux con las opciones `binary and quote site rdw`. Un archivo QSAM da soporte a todos los tipos de datos COBOL del registro.

*** nombre-datos calificado**
Un identificador que se compone de un nombre de datos seguido de uno o más conjuntos de cualquiera de las conexiones `OF` y `IN` seguido de un calificador de nombre de datos.

*** calificador**
(1) Un nombre de datos o un nombre asociado a un indicador de nivel que se utiliza en una referencia junto con otro nombre de datos (que es el nombre de un elemento que está subordinado al calificador) o junto con un nombre de condición. (2) Un nombre de sección que se utiliza en una referencia junto con un nombre de párrafo especificado en dicha sección. (3) Un nombre de biblioteca que se utiliza en una referencia junto con un nombre de texto asociado con esa biblioteca.

R

*** acceso aleatorio**
Modalidad de acceso en la que el valor especificado por el programa de un elemento de datos clave identifica el registro lógico obtenido, suprimido o colocado en un archivo relativo o indexado.

*** registro**
Véase *registro lógico*.

*** área de registro**
Área de almacenamiento asignada con el fin de procesar el registro descrito en una entrada de descripción de registro en el FILE SECTION del DATA DIVISION. En FILE SECTION, el número actual de posiciones de caracteres en el área de registro viene determinado por la cláusula RECORD explícita o implícita.

*** descripción de registro**
Véase *entrada de descripción de registro*.

*** entrada de descripción de registro**
Conjunto total de entradas de descripción de datos asociadas a un registro determinado. Sinónimo de *descripción de registro*.

Clave de registro
Clave cuyo contenido identifica un registro dentro de un archivo indexado.

nombre-clave-registro
Palabra definida por el usuario que nombra una clave asociada a un archivo indexado.

*** nombre-registro**
Palabra definida por el usuario que nombra un registro descrito en una entrada de descripción de registro en el DATA DIVISION de un programa COBOL.

*** número de registro**
El número ordinal de un registro en el archivo cuya organización es secuencial.

modalidad de grabación
El formato de los registros lógicos en un archivo. La modalidad de registro puede ser F (longitud fija), V (longitud variable), S (distribuida) o U (no definida).

recursividad
Programa que se llama a sí mismo o que es llamado directa o indirectamente por uno de sus programas llamados.

con capacidad recursiva
Un programa tiene capacidad recursiva (se puede llamar recursivamente) si el atributo RECURSIVE está en la sentencia PROGRAM-ID .

carrete
Parte discreta de un medio de almacenamiento, cuyas dimensiones son determinadas por cada implementador que contiene parte de un archivo, todo un archivo o cualquier número de archivos. Sinónimo de *unidad* y *volumen*.

reentrante
El atributo de un programa o rutina que permite que más de un usuario comparta una sola copia de un objeto de programa módulo de carga.

*** formato de referencia**
Formato que proporciona un método estándar para describir programas fuente COBOL.

Modificación de referencia
Método para definir un nuevo elemento de datos alfanumérico de categoría, DBCS de categoría o nacional de categoría especificando el carácter más a la izquierda y la longitud relativa a la posición de carácter más a la izquierda de un elemento de datos USAGE DISPLAY, DISPLAY-1o NATIONAL .

*** modificador-referencia**
Combinación sintácticamente correcta de series de caracteres y separadores que define un elemento de datos exclusivo. Incluye un separador de paréntesis izquierdo delimitador, la posición de carácter más a la izquierda, un separador de dos puntos, opcionalmente una longitud y un separador de paréntesis derecho delimitador.

*** relación**
Véase *operador relacional* o *condición de relación*.

*** carácter de relación**
Un carácter que pertenece al conjunto siguiente:

Carácter	Significado
>	Mayor que
<	Menor que
=	Igual a

*** condición de relación**
La proposición (para la que se puede determinar un valor de verdad) de que el valor de una expresión aritmética, elemento de datos, literal alfanumérico o nombre de índice tiene una relación específica con el valor de otra expresión aritmética, elemento de datos, literal alfanumérico o nombre de índice. Véase también *operador relacional*.

*** operador relacional**
Una palabra reservada, un carácter de relación, un grupo de palabras reservadas consecutivas o un grupo de palabras reservadas consecutivas y caracteres de relación utilizados en la construcción de una condición de relación. Los operadores permitidos y sus significados son:

Carácter	Significado
`IS GREATER THAN`	Mayor que
`IS >`	Mayor que
`IS NOT GREATER THAN`	No superior a
`IS NOT >`	No superior a
`IS LESS THAN`	Menor que
`IS <`	Menor que
`IS NOT LESS THAN`	No inferior a
`IS NOT <`	No inferior a
`IS EQUAL TO`	Igual a
`IS =`	Igual a
`IS NOT EQUAL TO`	No igual a
`IS NOT =`	No igual a
`IS GREATER THAN OR EQUAL TO`	Mayor que o igual a
`IS >=`	Mayor que o igual a
`IS LESS THAN OR EQUAL TO`	Menor que o igual a
`IS <=`	Menor que o igual a

*** archivo relativo**
Un archivo con una organización relativa.

*** clave relativa**
Clave cuyo contenido identifica un registro lógico en un archivo relativo.

*** organización relativa**
Estructura de archivo lógico permanente en la que cada registro se identifica de forma exclusiva mediante un valor entero mayor que cero, que especifica la posición ordinal lógica del registro en el archivo.

*** número de registro relativo**
El número ordinal de un registro en un archivo cuya organización es relativa. Este número se trata como un literal numérico que es un entero.

*** palabra reservada**
Palabra COBOL que se especifica en la lista de palabras que se pueden utilizar en un programa fuente COBOL, pero que no debe aparecer en el programa como palabra o nombre de sistema definido por el usuario.

*** recurso**
Recurso o servicio, controlado por el sistema operativo, que puede utilizar un programa en ejecución.

*** identificador resultante**
Elemento de datos definido por el usuario que debe contener el resultado de una operación aritmética.

rutina
Conjunto de sentencias de un programa COBOL que hace que el sistema realice una operación o serie de operaciones relacionadas.

*** nombre-rutina**
Palabra definida por el usuario que identifica un procedimiento escrito en un lenguaje distinto de COBOL.

Sistema de archivos RSD
El sistema de archivos delimitado secuencial de registros es un sistema de archivos de estación de trabajo que soporta archivos secuenciales. Un archivo RSD da soporte a todos los tipos de datos COBOL en registros de longitud fija o variable, puede ser editado por la mayoría de editores de archivos y puede ser leído por programas escritos en otros lenguajes. Este sistema sólo soporta archivos secuenciales.

*** tiempo de ejecución**
La hora a la que se ejecuta un programa objeto. Sinónimo de *hora de objeto*.

entorno de ejecución
Entorno en el que se ejecuta un programa COBOL.

*** unidad de ejecución**
Programa de objeto autónomo, o varios programas de objeto, que interactúan mediante sentencias COBOL `CALL` y funcionan en tiempo de ejecución como una entidad.

S

SBCS
Véase *juego de caracteres de un solo byte (SBCS)*.

terminador de ámbito
Palabra reservada de COBOL que marca el final de determinadas sentencias de `PROCEDURE DIVISION` statements.It puede ser explícita (`END-ADD`, por ejemplo) o implícita (punto separador).

*** sección**
Un conjunto de cero, uno o más párrafos o entidades, denominado un cuerpo de sección, el primero de los cuales va precedido de una cabecera de sección. Cada sección consta de la cabecera de sección y el cuerpo de sección relacionado.

*** cabecera de sección**
Combinación de palabras seguida de un punto de separación que indica el principio de una sección en cualquiera de estas divisiones: `ENVIRONMENT`, `DATA`o `PROCEDURE`. En `ENVIRONMENT DIVISION` y `DATA DIVISION`, una cabecera de sección se compone de palabras reservadas seguidas de un punto separador. Las cabeceras de sección permitidas en `ENVIRONMENT DIVISION` son:

```
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
```

Las cabeceras de sección permitidas en `DATA DIVISION` son:

```
FILE SECTION.
WORKING-STORAGE SECTION.
LOCAL-STORAGE SECTION.
LINKAGE SECTION.
```

En PROCEDURE DIVISION, una cabecera de sección se compone de un nombre de sección, seguido de la palabra reservada SECTION, seguida de un punto de separación.

*** nombre-sección**
Una palabra definida por el usuario que nombra una sección en PROCEDURE DIVISION.

segmentación

Hace referencia al módulo de segmentación estándar COBOL 85. Esta característica ha quedado obsoleta y se ha eliminado de las versiones posteriores de COBOL Standard y no está soportada en COBOL para Linux.

estructura de selección
Lógica de proceso de programa en la que se ejecuta una u otra serie de sentencias, en función de si una condición es verdadera o falsa.

*** frase**
Secuencia de una o más sentencias, la última de las cuales termina con un punto de separación.

*** programa compilado por separado**
Programa que, junto con sus programas contenidos, se compila por separado de todos los demás programas.

*** separador**
Un carácter o dos caracteres contiguos utilizados para delimitar series de caracteres.

*** coma de separador**
Una coma (,) seguida de un espacio utilizado para delimitar series de caracteres.

*** punto separador**
Un punto (.) seguido de un espacio utilizado para delimitar series de caracteres.

*** punto y coma de separador**
Un punto y coma (;) seguido de un espacio utilizado para delimitar las series de caracteres.

estructura de secuencia
Lógica de proceso de programa en la que se ejecuta una serie de sentencias en orden secuencial.

*** acceso secuencial**
Modalidad de acceso en la que los registros lógicos se obtienen de un archivo o se colocan en él en una secuencia de registro lógico de predecesor a sucesor consecutiva determinada por el orden de los registros del archivo.

*** archivo secuencial**
Un archivo con organización secuencial.

*** organización secuencial**
Estructura de archivo lógico permanente en la que un registro se identifica mediante una relación predecesor-sucesor establecida cuando el registro se coloca en el archivo.

búsqueda en serie
Búsqueda en la que se examinan consecutivamente los miembros de un conjunto, empezando por el primer miembro y terminando por el último.

Sistema de archivos SFS
El sistema de archivos CICS Structured File Server es un sistema de archivos orientado a registros que da soporte al acceso secuencial, relativo y a archivos indexados por clave.

biblioteca compartida
Biblioteca creada por el enlazador que contiene al menos una subrutina que pueden utilizar varios procesos. Los programas y subrutinas están enlazados como de costumbre, pero el código común a diferentes subrutinas se combina en un archivo de biblioteca que se puede cargar en tiempo de ejecución y compartir con muchos programas. Una clave para identificar el archivo de biblioteca compartida está en la cabecera de cada subrutina.

*** condición de signo**
La proposición (para la que se puede determinar un valor de verdad) de que el valor algebraico de un elemento de datos o una expresión aritmética es menor que, mayor que o igual a cero.

firma
El nombre de una operación y sus parámetros.

*** condición simple**
Cualquier condición individual elegida de este conjunto:

- Condición de relación
- Condición de clase
- Condición de nombre de condición
- Condición de estado de conmutador
- Condición de firma

Véase también *condición* y *condición simple negada*.

juego de caracteres de un solo byte (SBCS)
Conjunto de caracteres en el que cada carácter se representa mediante un solo byte. Consulte también *ASCII* y *EBCDIC (Extended Binary-Coded Decimal Interchange Code)*.

bytes de holgura (dentro de registros)
Bytes insertados por el compilador entre elementos de datos para garantizar la alineación correcta de algunos elementos de datos elementales. Los bytes de holgura no contienen datos significativos. La cláusula SYNCHRONIZED indica al compilador que inserte bytes de holgura cuando sean necesarios para una alineación adecuada.

bytes de holgura (entre registros)
Bytes insertados por el programador entre registros lógicos bloqueados de un archivo, para garantizar la alineación correcta de algunos elementos de datos elementales. En algunos casos, los bytes de holgura entre registros mejoran el rendimiento de los registros procesados en un almacenamiento intermedio.

*** ordenar archivo**
Colección de registros que se ordenarán mediante una sentencia SORT . El archivo de ordenación se crea y sólo puede ser utilizado por la función de ordenación.

*** entrada de descripción de archivo de fusión de clasificación**
Una entrada en FILE SECTION del DATA DIVISION que se compone del indicador de nivel SD, seguido de un nombre de archivo y, a continuación, seguido de un conjunto de cláusulas de archivo según sea necesario.

*** SOURCE-COMPUTER**
El nombre de un párrafo ENVIRONMENT DIVISION en el que se describe el entorno del sistema, donde se compila el programa fuente.

*** entrada de sistema de origen**
Una entrada en el párrafo SOURCE-COMPUTER de ENVIRONMENT DIVISION; esta entrada contiene cláusulas que describen el entorno del sistema en el que se va a compilar el programa fuente.

*** elemento de origen**
Identificador designado por una cláusula SOURCE que proporciona el valor de un elemento imprimible.

programa fuente
Aunque un programa fuente puede estar representado por otras formas y símbolos, en este documento el término siempre hace referencia a un conjunto sintácticamente correcto de sentencias COBOL. Un programa fuente COBOL comienza con la sentencia IDENTIFICATION DIVISION o COPY y termina con el marcador de programa final, si se especifica, o con la ausencia de líneas de programa fuente adicionales.

unidad de origen
Unidad de código fuente COBOL que se puede compilar por separado: un programa. También se conoce como *unidad de compilación*.

Carácter especial
Un carácter que pertenece al siguiente conjunto:

Carácter	Significado
+	Signo más
-	Signo menos (guión)
*	Asterisco
/	Inclinado (barra inclinada)
=	Signo igual
$	Símbolo de moneda
,	Coma
;	Punto y coma
.	Punto (coma decimal, punto completo)
"	comilla
'	Apóstrofo
(	Paréntesis izquierdo
)	Paréntesis derecho
>	Mayor que
<	Menor que
:	Dos puntos
_	Subrayado

SPECIAL-NAMES
El nombre de un párrafo de ENVIRONMENT DIVISION en el que los nombres de entorno están relacionados con nombres mnemotécnicos especificados por el usuario.

*** entrada de nombres especiales**
Una entrada en el párrafo SPECIAL-NAMES del ENVIRONMENT DIVISION; esta entrada proporciona medios para especificar el signo de moneda; elegir el separador decimal; especificar caracteres simbólicos; relacionar nombres de implementador con nombres mnemónicos especificados por el usuario; relacionar nombres de alfabeto con conjuntos de caracteres o secuencias de clasificación; y relacionar nombres de clase con conjuntos de caracteres.

*** registros especiales**
Determinadas áreas de almacenamiento generadas por el compilador cuyo uso principal es almacenar información producida junto con el uso de una característica COBOL específica.

*** sentencia**
Combinación sintácticamente válida de palabras, literales y separadores, empezando por un verbo, escrito en un programa fuente COBOL.

Sistema de archivos STL
El sistema de archivos de lenguaje estándar es el sistema de archivos de estación de trabajo nativa para COBOL. Este sistema soporta archivos secuenciales, relativos e indexados.

programación estructurada
Una técnica para organizar y codificar un programa de computadora en el cual el programa comprende una jerarquía de segmentos, cada segmento tiene un único punto de entrada y un único punto de salida. El control se pasa hacia abajo a través de la estructura sin ramas incondicionales a niveles más altos de la jerarquía.

*** asunto de entrada**
Un operando o palabra reservada que aparece inmediatamente después del indicador de nivel o el número de nivel en una entrada de DATA DIVISION .

*** subprograma**
Véase *programa llamado*.

*** subíndice**
Número de aparición representado por un entero, un nombre de datos seguido opcionalmente por un entero con el operador + o-, o un nombre de índice seguido opcionalmente por un entero con el operador + o-, que identifica un elemento determinado de una tabla. Un subíndice puede ser la palabra ALL cuando se utiliza el identificador de subíndice como argumento de función para una función que permite un número variable de argumentos.

*** nombre-datos-subíndice**
Identificador compuesto de un nombre de datos seguido de uno o más subíndices entre paréntesis.

Carácter de sustitución
Carácter que se utiliza en una conversión de una página de códigos fuente a una página de códigos de destino para representar un carácter que no está definido en la página de códigos de destino.

par sustituto
En el formato UTF-16 de Unicode, un par de unidades de codificación que juntas representan un único carácter gráfico Unicode. La primera unidad del par se denomina *sustituto alto* y la segunda *sustituto bajo*. El valor de código de un sustituto alto está en el rango de X'D800'a X'DBFF'. El valor de código de un sustituto bajo está en el rango de X'DC00'a X'DFFF'. Los pares suplentes proporcionan más caracteres que los 65.536 caracteres que caben en el juego de caracteres codificado Unicode de 16 bits.

condición de estado de conmutador
La proposición (para la cual se puede determinar un valor de verdad) de que un conmutador UPSI, capaz de establecerse en un estado de encendido o apagado, se ha establecido en un estado específico.

*** carácter-simbólico**
Palabra definida por el usuario que especifica una constante figurativa definida por el usuario.

Sintaxis
(1) La relación entre caracteres o grupos de caracteres, independientemente de su significado o de la forma en que se interpreten y utilicen. (2) La estructura de las expresiones en un lenguaje. (3) Las normas que rigen la estructura de una lengua. (4) La relación entre símbolos. (5) Normas para la construcción de una declaración.

SYSADATA
Archivo de información de compilación adicional que se genera si la opción de compilador ADATA está en vigor.

SYSIN
El archivo o archivos de entrada del compilador primario.

syslib
El archivo o archivos de entrada del compilador secundario, que se procesan si la opción de compilador LIB está en vigor.

SYSPRINT
El archivo de listado del compilador.

*** nombre-sistema**
Palabra COBOL que se utiliza para comunicarse con el entorno operativo.

T

*** tabla**
Un conjunto de elementos de datos lógicamente consecutivos que se definen en DATA DIVISION mediante la cláusula OCCURS .

*** elemento de tabla**
Elemento de datos que pertenece al conjunto de elementos repetidos que forman una tabla.

*** nombre-texto**
Palabra definida por el usuario que identifica el texto de la biblioteca.

*** palabra de texto**
Un carácter o una secuencia de caracteres contiguos entre el margen A y el margen R en una biblioteca COBOL, programa fuente o pseudotexto que sea cualquiera de los caracteres siguientes:

- Un separador, excepto para el espacio; un delimitador de pseudotexto; y los delimitadores de apertura y cierre para literales alfanuméricos. Los caracteres de paréntesis derecho y paréntesis izquierdo, independientemente del contexto dentro de la biblioteca, programa fuente o pseudotexto, siempre se consideran palabras de texto.
- Un literal que incluye, en el caso de literales alfanuméricos, la comilla de apertura y la comilla de cierre que enlazan el literal.
- Cualquier otra secuencia de caracteres COBOL contiguos excepto las líneas de comentario y la palabra C0PY limitada por separadores que no son ni un separador ni un literal.

hebra
Secuencia de instrucciones de sistema (iniciadas por una aplicación dentro de un proceso) que está en control de un proceso.

señal
En el editor COBOL, unidad de significado de un programa. Una señal puede contener datos, una palabra clave de idioma, un identificador u otra parte de la sintaxis de lenguaje.

diseño descendente
Diseño de un programa informático que utiliza una estructura jerárquica en la que se realizan funciones relacionadas en cada nivel de la estructura.

Desarrollo descendente
Véase *programación estructurada*.

etiqueta-remolque
(1) Una etiqueta de que sigue a los registros de datos en una unidad de soporte de grabación. (2) Sinónimo de *etiqueta de fin de archivo*.

resolución de problemas
Para detectar, localizar y eliminar problemas en el uso de software de computadora.

*** valor de verdad**
La representación del resultado de la evaluación de una condición en términos de uno de dos valores: verdadero o falso.

U

*** operador unario**
Un signo más (+) o un signo menos (-) que precede a una variable o un paréntesis izquierdo en una expresión aritmética y que tiene el efecto de multiplicar la expresión por + 1 o -1, respectivamente.

Unicode
Estándar de codificación de caracteres universal que da soporte al intercambio, proceso y visualización de texto escrito en cualquiera de los idiomas del mundo moderno. Existen varios esquemas de codificación para representar Unicode, incluidos UTF-8, UTF-16y UTF-32. COBOL para Linux da soporte a Unicode utilizando UTF-16 en formato little-endian como representación para el tipo de datos nacional.

Identificador uniforme de recursos (URI)
Secuencia de caracteres que nombra de forma exclusiva un recurso; en COBOL para Linux, el identificador de un espacio de nombres. La sintaxis de URI la define el documento *Identificador universal de recursos (URI): Sintaxis genérica*.

unit
Un módulo de acceso directo, cuyas dimensiones están determinadas por IBM.

*** ejecución no satisfactoria**
El intento de ejecución de una sentencia que no da como resultado la ejecución de todas las operaciones especificadas por dicha sentencia. La ejecución no satisfactoria de una sentencia no afecta a los datos a los que hace referencia dicha sentencia, pero puede afectar a los indicadores de estado.

Conmutador UPSI
Conmutador de programa que realiza las funciones de un conmutador de hardware. Se proporcionan ocho: UPSI-0 a UPSI-7.

URI
Véase *Uniform Resource Identifier (URI)*.

*** palabra definida por el usuario**
Palabra COBOL que debe proporcionar el usuario para satisfacer el formato de una cláusula o sentencia.

V

*** variable**
Elemento de datos cuyo valor se puede cambiar mediante la ejecución del programa objeto. Una variable utilizada en una expresión aritmética debe ser un elemento elemental numérico.

elemento de longitud variable
Elemento de grupo que contiene una tabla descrita con la frase `DEPENDING` de la cláusula `OCCURS` .

*** registro de longitud variable**
Un registro asociado a un archivo cuya descripción de archivo o entrada de descripción de fusión de clasificación permite que los registros contengan un número variable de posiciones de caracteres.

*** elemento de datos de aparición variable**
Un elemento de datos de aparición de variable es un elemento de tabla que se repite un número variable de veces. Un elemento de este tipo debe contener una cláusula `OCCURS DEPENDING ON` en su entrada de descripción de datos o estar subordinado a dicho elemento.

*** grupo de ubicación variable**
Elemento de grupo que sigue, y no subordinado, a una tabla de longitud variable en el mismo registro. El elemento de grupo puede ser un grupo alfanumérico o un grupo nacional.

*** elemento de ubicación variable**
Un elemento de datos a continuación, y no subordinado, de una tabla de longitud variable en el mismo registro.

*** verbo**
Palabra que expresa una acción que debe realizar un compilador COBOL o un programa de objetos.

volumen
Un módulo de almacenamiento externo. Para los dispositivos de cinta es un carrete; para los dispositivos de acceso directo es una unidad.

Sistema de archivos VSAM
Sistema de archivos que da soporte a organizaciones COBOL secuenciales, relativas e indexadas.

VSAM
Término genérico para el *sistema de archivos STL* o el *sistema de archivos SFS*.

W

servicio web
Una aplicación modular que realiza tareas específicas y es accesible a través de protocolos abiertos como HTTP y SOAP.

espacio en blanco
Caracteres que introducen espacio en un documento. Son las siguientes:

- Espacio
- Tabulación horizontal
- Retorno de carro
- Salto de línea
- Línea siguiente

tal como se indica en el estándar Unicode.

campo de fecha de ventana
Campo de fecha que contiene un año con ventana (dos dígitos). Véase también *campo de fecha* y *año de ventana*.

año de ventana
Campo de fecha que consta sólo de un año de dos dígitos. Este año de dos dígitos se puede interpretar utilizando una ventana de siglo. Por ejemplo, 10 podría interpretarse como 2010. Véase también *ventana de siglo*. Compárese con *año expandido*.

*** palabra**
Serie de caracteres de no más de 30 caracteres que forma una palabra definida por el usuario, un nombre de sistema, una palabra reservada o un nombre de función.

*** WORKING-STORAGE SECTION**
La sección del DATA DIVISION que describe los elementos de datos de WORKING-STORAGE , compuestos de elementos no contiguos o registros de WORKING-STORAGE o de ambos.

estación de trabajo
Término genérico para sistemas, incluidos sistemas personales, terminales 3270, estaciones de trabajo inteligentes y terminales UNIX . A menudo, una estación de trabajo está conectada a un sistema principal o a una red.

envoltura
Objeto que proporciona una interfaz entre el código orientado a objetos y el código orientado a procedimientos. El uso de derivadores permite que otros sistemas reutilicen y accedan a los programas.

X

X
El símbolo de una cláusula PICTURE que puede contener cualquier carácter del juego de caracteres del sistema.

XML
Extensible Markup Language. Un metalenguaje estándar para definir lenguajes de marcación que se ha derivado de y es un subconjunto de SGML. XML omite las partes más complejas y menos utilizadas de SGML y hace que sea mucho más fácil escribir aplicaciones para manejar tipos de documentos, crear y gestionar información estructurada, y transmitir y compartir información estructurada entre diversos sistemas informáticos. El uso de XML no requiere las aplicaciones robustas y el proceso necesario para SGML. XML se desarrolla bajo los auspicios de World Wide Web Consortium (W3C).

Datos XML
Datos que se organizan en una estructura jerárquica con elementos XML. Las definiciones de datos se definen en las declaraciones de tipo de elemento XML.

declaración XML
Texto XML que especifica las características del documento XML como, por ejemplo, la versión de XML que se está utilizando y la codificación del documento.

Documento XML
Un objeto de datos que está bien formado tal como lo define la especificación XML W3C .

Espacio de nombres XML
Mecanismo, definido por las especificaciones de espacio de nombres XML W3C , que limita el ámbito de una colección de nombres de elemento y nombres de atributo. Un espacio de nombres XML elegido de forma exclusiva garantiza la identidad exclusiva de un nombre de elemento o nombre de atributo en varios documentos XML o varios contextos dentro de un documento XML.

S

expansión de campo de año
Expansión explícita de campos de fecha que contienen años de dos dígitos para contener años de cuatro dígitos en archivos y bases de datos y, a continuación, uso de estos campos en formato expandido en programas. Este es el único método para asegurar un proceso de fecha fiable para las aplicaciones que han utilizado años de dos dígitos.

Z

elemento de datos decimales con zona
Elemento de datos decimal externo que se describe implícita o explícitamente como USAGE DISPLAY y que contiene una combinación válida de PICTURE símbolos 9, S, Py V. El contenido de un elemento de datos decimal con zona se representa en caracteres del 0 al 9, opcionalmente con un signo. Si la serie PICTURE especifica un signo y se especifica la cláusula SIGN IS SEPARATE , el signo se representa como caracteres + o-. Si no se especifica SIGN IS SEPARATE , el signo es un dígito hexadecimal que se superpone a los 4 primeros bits de la posición del signo (inicial o final).

#

entrada-descripción-nivel-77
Entrada de descripción de datos que describe un elemento de datos no contiguo que tiene el número de nivel 77.

85 COBOL Estándar
El lenguaje COBOL definido por los estándares siguientes:

- *ANSI INCITS 23-1985, lenguajes de programación-COBOL*, modificado por *ANSI INCITS 23a-1989, Lenguajes de programación-COBOL-Módulo de función intrínseca para COBOL*
- *ISO 1989:1985, Lenguajes de programación-COBOL*, modificado por *ISO/IEC 1989/AMD1:1992, Lenguajes de programación-COBOL: Módulo de función intrínseca*

Estándar COBOL 2002
El lenguaje COBOL definido por el siguiente estándar:

- *INCITS/ISO/IEC 1989-2002, Tecnologías de la información-Lenguajes de programación-COBOL*

Estándar COBOL 2014
El lenguaje COBOL definido por el siguiente estándar:

- *INCITS/ISO/IEC 1989:2014, Tecnologías de la información-Lenguajes de programación, sus entornos e interfaces de software del sistema-Lenguaje de programación COBOL*

Lista de recursos

Publicaciones de COBOL for Linux on x86

Novedades en COBOL for Linux en x86 1.2, SC28-3453-00

Guía de instalación, GC28-3116-01

Consulta de lenguaje, SC28-3117-01

Guía de programación, SC28-3118-01

Guía de migración, SC28-3454-00

Apoyo

Si tiene un problema al utilizar COBOL for Linux, Visite el sitio web IBM Support , que proporciona información de soporte actualizada.

Publicaciones relacionadas

DB2 para Linux, UNIX, y Windows

Puede encontrar las publicaciones siguientes en la Documentación deIBM Documentation:

- *Consulta de mandatos*
- *Conceptos de administración de bases de datos y referencia de configuración*
- *Referencia de SQL para Db2 Versión 11.1 para Linux, UNIXy Windows*

TXSeries para Multiplatforms

- *Documentación deIBM TXSeries for Multiplatforms*

IBM CICS TX

- *IBM Documentación de CICS TX*

Representación de Unicode y caracteres

- *Unicode*
- *Componentes internacionales para Unicode: Converter Explorer*
- *Arquitectura de representación de datos de caracteres: Referencia y registro*

XML

- *XML (Extensible Markup Language)*
- *Espacios de nombres en XML 1.0*
- *Espacios de nombres en XML 1.1*
- *Especificación XML*

Índice

Caracteres Especiales

Números

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

Z

Número de Programa: 5737-L11

SC28-3118-01