# *Guía de migración*

SC28-3454-00

**Nota**

Antes de utilizar esta información y el producto al que da soporte, lea la información general que se encuentra en el apartado “Avisos” en la página 47.

**Primera edición (enero de 2023)**

Esta edición se aplica a IBM® COBOL for Linux® en x86 1.2 o compiladores 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

# Capítulo 1. Introducción

Este documento presenta temas para ayudarle a migrar programas fuente COBOL a COBOL for Linux en x86 1.2.

Si está actualizando desde una versión anterior de COBOL for Linux en x86, consulte Capítulo 2, “Migración de una versión anterior de COBOL for Linux en x86 a la versión actual”, en la página 3 para obtener información sobre los cambios que debe tener en cuenta al compilar y ejecutar las aplicaciones COBOL con COBOL for Linux en x86 1.2.

COBOL for Linux en x86 forma parte de la familia de compilador COBOL IBM . La migración de programas COBOL desde Enterprise COBOL for z/OS o COBOL for AIX a COBOL for Linux en x86 debe ser muy sencilla. Sin embargo, es posible que tenga que actualizar algunas de las opciones de compilador y variables de entorno y tener en cuenta las diferencias en la representación de datos entre las plataformas. Desde una perspectiva de lenguaje COBOL, todos los compiladores COBOL de IBM se ajustan al estándar ISO 1989:1985, *Lenguajes de programación-COBOL* , por lo tanto, la mayoría del código fuente COBOL se debe compilar correctamente con COBOL for Linux en x86. Enterprise COBOL en z/OS es el más avanzado de los compiladores IBM COBOL y da soporte a varias características en los estándares de lenguaje COBOL 2002 y COBOL 2014. Si está migrando desde Enterprise COBOL for z/OS 6 y el origen COBOL contiene el uso de nuevas características de lenguaje, consulte “Elementos de lenguaje” en la página 34 para confirmar si estas características están disponibles en COBOL for Linux en x86. Para obtener más información, consulte Capítulo 4, “Migración de Enterprise COBOL for z/OS a COBOL for Linux en x86”, en la página 29 y Capítulo 5, “Migración de COBOL for AIX a COBOL for Linux en x86”, en la página 43.

Los programas COBOL creados utilizando compiladores COBOL que no son deIBM normalmente tendrán algunas diferencias de código fuente que deberán modificarse para que sean compatibles con COBOL for Linux en x86. Estas diferencias suelen ser el resultado de extensiones o desviaciones de implementador de COBOL estándar, pero también pueden deberse a elementos del estándar COBOL que se han especificado como "implementador definido". El esfuerzo para migrar a COBOL for Linux en x86 dependerá en gran medida de cuántas diferencias con COBOL estándar se hayan codificado en los archivos de origen COBOL que está migrando. COBOL for Linux en x86 incluye un programa de utilidad de conversión de origen (scu) para ayudar con este esfuerzo de migración. Es posible que también sea necesario tener en cuenta las diferencias en la representación de datos entre las plataformas y los compiladores COBOL. Para obtener más información, consulte Capítulo 3, “Migración de compiladores COBOL que no son deIBM a COBOL for Linux en x86”, en la página 7.

En este documento, el término COBOL 2002 hace referencia al estándar de lenguaje ISO/IEC 1989:2002, *Information technology-Programming languages-COBOL* . Esto no implica que COBOL for Linux en x86 haya implementado las nuevas características que se encuentran en COBOL 2002, aunque algunas se hayan implementado. Simplemente estamos utilizando COBOL 2002 para ayudarle a identificar las extensiones de COBOL que puede encontrar a medida que realiza la migración, y para ayudarle a determinar los cambios que puede necesitar realizar para resolver las extensiones que encuentre. Le resultará útil consultar la publicación COBOL for Linux en x86 *Language Reference* y la publicación *Programming Guide* al modificar el código COBOL. Puede encontrar estos documentos en la documentación deCOBOL for Linux en x86.

# Capítulo 2. Migración de una versión anterior de COBOL for Linux en x86 a la versión actual

Esta información describe algunas áreas que es posible que deba tener en cuenta si migra programas de COBOL for Linux en x86 1.1 a 1.2.

## Cambios de opción

Esta información describe los cambios en las opciones de COBOL for Linux en x86 1.2.

**ADDR**
La opción predeterminada se cambia de `ADDR(32)` a `ADDR(64)`. El cambio significa que si desea generar un programa de 32 bits en COBOL for Linux en x86 1.2, debe especificar explícitamente `ADDR(32)`, de lo contrario, generará un programa de 64 bits.

## Migración de aplicaciones de 32 bits a la modalidad de 64 bits

Los resultados de ejecución pueden cambiar en modalidad de 64 bits. Si migra aplicaciones de 32 bits a la modalidad de 64 bits, es posible que tenga que realizar alguna recodificación para acomodar las diferencias.

Aunque la principal consideración en la migración a la modalidad de 64 bits es el cambio en el tamaño de los elementos de datos, otras áreas de una aplicación podrían verse potencialmente afectadas, tal como se describe a continuación.

**Registro especial de `LENGTH OF` :**

En la modalidad de 32 bits, la definición implícita del registro especial `LENGTH OF` es `PICTURE 9(9) USAGE IS BINARY`. En la modalidad de 64 bits, la definición implícita es `PICTURE 9(18) USAGE IS BINARY`.

**Función intrínseca de `LENGTH` :**

En modalidad de 32 bits, `FUNCTION LENGTH` devuelve un entero de 9 dígitos. En modalidad de 64 bits, devuelve un entero de 18 dígitos.

**Elementos de datos de dirección y elementos de datos de índice:**

La asignación de almacenamiento para elementos de datos que contienen direcciones o índices es de 4 bytes en modalidad de 32 bits y de 8 bytes en modalidad de 64 bits. Los elementos de datos que se ven afectados son los que están definidos con cualquiera de los usos siguientes:

- `POINTER`
- `FUNCTION-POINTER`
- `PROCEDURE-POINTER`
- `INDEX`

El registro especial `ADDRESS OF` está definido implícitamente como `USAGE POINTER`. Por lo tanto, el almacenamiento se asigna para él tal como se describe anteriormente para los elementos de datos de `USAGE POINTER` .

**Diseño de elementos de grupo:**

El cambio en el tamaño de asignación de elementos de datos de puntero y elementos de datos de índice entre programas de modalidad de 32 bits y de modalidad de 64 bits, descrito anteriormente, afecta al diseño de elementos de grupo. En concreto, los elementos de datos que se encuentran después de un elemento de datos de puntero o índice podrían estar ubicados en diferentes desplazamientos dentro de

un grupo. Además,el número de *bytes de holgura* (bytes insertados por el compilador para garantizar una alineación correcta) podría diferir y, por lo tanto, el tamaño total de un grupo podría cambiar.

**Restricción:** Los programas COBOL no deben basarse en el diseño de elementos de grupo.

**Elementos de datos de SYNCHRONIZED :**

Si se especifica la cláusula SYNCHRONIZED para elementos de datos de puntero o elementos de datos de índice, los elementos se alinean en un límite de palabra completa en modalidad de 32 bits y en un límite de palabra doble en modalidad de 64 bits.

Puesto que el tamaño de los punteros es diferente entre las aplicaciones de 32 bits y de 64 bits, los desplazamientos de los elementos de datos que son posteriores a ellos en un grupo, así como el tamaño global del grupo, cambiarán. Esto es cierto independientemente de si el compilador inserta o no bytes de holgura cuando se especifica SYNCHRONIZED . Sin embargo, especificar SYNCHRONIZED afectará aún más a los desplazamientos de los punteros y los elementos de datos subsiguientes de un grupo, así como al tamaño del grupo debido a la diferencia en la alineación de los punteros entre aplicaciones de 32 bits y 64 bits.

**Redefinición de elementos de datos de puntero:**

Debido a que el tamaño de un puntero es diferente entre aplicaciones de 32 bits y de 64 bits, su uso como objeto de una cláusula REDEFINES podría producir resultados no deseados a menos que el tamaño del elemento de datos de redefinición (es decir, el sujeto REDEFINES) también se cambie en consecuencia. Sin embargo, incluso si cambia el elemento de datos de redefinición para que coincida con el tamaño del puntero, considere cuidadosamente por qué está redefiniendo un puntero en primer lugar. El valor de un puntero es una dirección y, en general, el programa sólo debe utilizar punteros en las sentencias COBOL que soportan explícitamente el uso de punteros como la sentencia SET. No se recomienda utilizar una cláusula REDEFINES para inspeccionar o establecer el valor de un puntero.

**Utilización de la variable de compilación condicional IGY-ADDR**

En general, se recomienda que codifique los programas para que no se vean afectados por el cambio de aplicaciones de 32 bits a 64 bits. Por ejemplo, no hay ninguna dependencia en el diseño de los elementos de datos de un grupo si codifica los programas de esta forma.

Sin embargo, si por alguna razón los programas son sensibles a la modalidad de direccionamiento, hay una variable de compilación condicional predefinida denominada IGY-ADDR, que se puede utilizar para tener distintas vías de acceso de código compiladas basándose en el valor de la opción ADDR. Esto le permite mantener un único archivo fuente que funcionará tanto para compilaciones de 32 bits como de 64 bits.

**Archivos de datos:**

Los archivos de datos creados por programas en modalidad de 32 bits pueden ser procesados por programas en modalidad de 64 bits, y los archivos de datos creados por programas en modalidad de 64 bits pueden ser procesados por programas en modalidad de 32 bits. No se adjunta ninguna modalidad de direccionamiento a un archivo de datos. Sin embargo, si una definición de registro contiene un elemento de datos de dirección o un elemento de datos de índice, la migración a la modalidad de 64 bits puede hacer que cambie la longitud del registro y, por lo tanto, la longitud del archivo.

**Subsistemas de middleware:**

Si un programa depende de un subsistema de middleware para realizar tareas de proceso de datos, asegúrese de que dicho subsistema funciona con programas de 64 bits.

**Restricción:** CICS TXSeries sólo da soporte a aplicaciones COBOL de 32 bits.

DB2 Versión 11.5 da soporte a aplicaciones COBOL de 32 bits o de 64 bits.

**Otras consideraciones:**

Linux no da soporte a la combinación de programas en modalidad de 32 bits y en modalidad de 64 bits dentro de una aplicación, y el enlazador no da soporte al enlace de archivos de objeto de distintas modalidades de direccionamiento. Si un programa de 64 bits realiza llamadas interlingüísticas, o llama a

rutinas en una biblioteca de usuario, asegúrese de que las versiones de 64 bits de las rutinas llamadas estén disponibles.

## Soporte de distribución de Linux

Las distribuciones de Linux que han alcanzado el final de soporte como, por ejemplo, RHEL 7.8 no están soportadas con COBOL for Linux en x86 1.2. Consulte la *Guía de instalación* para obtener una lista de distribuciones de Linux soportadas por COBOL for Linux en x86 1.2.

## Soporte de programas opcionales

Los programas opcionales que han alcanzado el final de soporte, como por ejemplo Db2 11.1, no están soportados con COBOL for Linux en x86 1.2. Consulte la *Guía de instalación* para obtener una lista de programas opcionales soportados por COBOL for Linux en x86 1.2.

## Compatibilidad de tiempo de ejecución

Al instalar COBOL for Linux en x86 1.2, la biblioteca de tiempo de ejecución se instalará en la ubicación `/opt/ibm/cobol/rte`, actualizando las versiones anteriores de la biblioteca de tiempo de ejecución que podrían haberse instalado en dicha ubicación. Puesto que la biblioteca de tiempo de ejecución COBOL es compatible con versiones superiores, los programas COBOL creados utilizando COBOL for Linux en x86 1.1 continuarán ejecutándose con la biblioteca de tiempo de ejecución proporcionada en COBOL for Linux en x86 1.2.

# Capítulo 3. Migración de compiladores COBOL que no son deIBM a COBOL for Linux en x86

Esta información describe algunas áreas que es posible que deba tener en cuenta si migra programas desde compiladores que no son deIBM COBOL a COBOL for Linux en x86.

## Cambios de formato

Para ejecutar los programas fuente en IBM COBOL for Linux en x86, realice los cambios de formato necesarios en los programas fuente.

### Sufijo de archivo .cob para origen COBOL

Ahora puede compilar un programa COBOL utilizando un sufijo `.cob` además del sufijo `.cbl` .

IBM COBOL for Linux en x86 da soporte al sufijo `.cob` para archivos de origen COBOL.

Consulte el siguiente ejemplo:

```
$ cob2 -o tcCobol tcCobol.cob
IBM COBOL para Linux 1.2.0 compile started
End of compilation 1, program TCCOBOL, no statements flagged.
```

### Archivos fuente COBOL de formato libre

IBM COBOL for Linux en x86 suprime algunos mensajes relacionados con la aplicación del código fuente de formato fijo.

Cuando utilice IBM COBOL for Linux en x86 para compilar código fuente que contenga código fuente COBOL de formato libre, recibirá los mensajes de error en el ejemplo siguiente:

```
$ cat tcFreeFormat.cbl
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. FreeFormat.
000300 ENVIRONMENT DIVISION.
000400 DATA DIVISION.
000500
000600 PROCEDURE DIVISION.
000700 BEGIN-MY-PROGRAM.
000800 DISPLAY "COBOL for Linux en x86 1.2.0...".
000900 STOP RUN.

$ cob2 tcFreeFormat.cbl
IBM COBOL para Linux 1.2.0 compile started
0LineID  Message code  Message text
      8  IGYPS0009-E   "DISPLAY" should not begin in area "A".  It was processed as if found
                       in area "B".
      9  IGYPS0009-E   "STOP" should not begin in area "A".  It was processed as if found in
                       area "B".
-Messages    Total    Informational    Warning    Error    Severe    Terminating
0Printed:        2                                  2
End of compilation 1,  program FREEFORMAT,  highest severity: Error.
Return code 8
```

Puede volver a escribir el origen en formato fijo o utilizar `scu` para convertir el origen de formato libre a origen de formato fijo:

```
$ cat sol-tcFreeFormat.cbl
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. FreeFormat.
000300 ENVIRONMENT DIVISION.
000400 DATA DIVISION.
000500
000600 PROCEDURE DIVISION.
000700 BEGIN-MY-PROGRAM.
```

```
000800     DISPLAY "COBOL for Linux en x86 1.2.0...".
000900     STOP RUN.
```

## Literales alfanuméricos de más de 160 caracteres en archivos de origen COBOL

La longitud de un literal alfanumérico, excluidos los separadores que delimitan el literal, debe ser mayor que cero y menor o igual que 160 posiciones de caracteres alfanuméricos.

Cuando utilice IBM COBOL for Linux en x86 para compilar el código fuente que contiene uno o más literales alfanuméricos de más de 160 caracteres, recibirá el mensaje de error en el ejemplo siguiente:

```
$ cat tcAlphanumeric160.cbl
   000100 IDENTIFICATION DIVISION.
   000200 PROGRAM-ID. TEST-CASE.
   000300 ENVIRONMENT DIVISION.
   000400
   000500 DATA DIVISION.
   000600 WORKING-STORAGE SECTION.
   000700
   000800 01 VARIABLE PIC N(300) VALUE "BEGIN = = = = = = =
   000900-       " = = = = = = = = = = = = = = = = = = = = =
   001000-       " = = = = = = = = = = = = = = = = = = = = =
   001100-       " = = = = = = = = = = = = = = = = = = = = =
   001200-       "= = = = = = = = = = = = = = END".
   001300
   001400 PROCEDURE DIVISION.
   001500     DISPLAY VARIABLE.
   001600     STOP RUN.

$ cob2 -o tcAlphanumeric160 tcAlphanumeric160.cbl IBM COBOL para Linux 1.2.0 compile started
0LineID  Message code  Message text
      8  IGYDS0026-E   An alphanumeric literal that was longer than 160 characters was
                       specified.  The literal was truncated to 160 characters.
-Messages    Total    Informational    Warning    Error    Severe    Terminating
0Printed:       1                                   1
End of compilation 1,  program TEST-CASE,  highest severity: Error.
Return code 8
```

Según COBOL 2002 estándar, la longitud de un literal alfanumérico, excluyendo los separadores que delimitan el literal, debe ser mayor que cero y menor o igual que 160 posiciones de caracteres alfanuméricos.

Cambie el código fuente COBOL para que siga a COBOL 2002 estándar. Puede cambiar el código para dividir literales largos. Tal como se muestra en el ejemplo siguiente, `MOVE “longliteral” to ITEM-1` se puede dividir:

```
MOVE “long” to ITEM-1(1:4).
MOVE “literal” to ITEM-1(5:).
```

## Periodos adicionales y fuera de lugar en el origen COBOL

Debe eliminar los puntos adicionales o mal colocados en el origen de formato fijo.

Cuando se utiliza IBM COBOL for Linux en x86 para compilar código fuente que contiene puntos adicionales y fuera de lugar, recibirá los mensajes de error en el ejemplo siguiente:

```
$ cat tcPeriods.cbl
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. TEST-CASE.
000030 ENVIRONMENT DIVISION.
000040 DATA DIVISION.
000050
000060 PROCEDURE DIVISION.
000070     PERFORM GREETING.
000080    .
000090
000100     STOP RUN.
000110
000120 GREETING.
000130 .
```

```
000140
000150     DISPLAY "HELLO FROM IBM.".

$ cob2 -o tcPeriods tcPeriods.cbl
IBM COBOL para Linux 1.2.0 compile started
0LineID  Message code  Message text
      8  IGYPS0019-W   No COBOL statement was found between periods.
                       Same message on line:     13
-Messages    Total    Informational    Warning    Error    Severe    Terminating
0Printed:       2                          2
End of compilation 1,  program TEST-CASE,  highest severity: Warning.
Return code 4
```

Según COBOL 2002 estándar, el origen de formato fijo no puede incluir puntos adicionales o ausentes en el origen.

Cambie el código fuente COBOL para que siga a COBOL 2002 estándar, o utilice `scu` para eliminar puntos adicionales y fuera de lugar:

```
$ cat sol-tcPeriods.cbl
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. TEST-CASE.
000030 ENVIRONMENT DIVISION.
000040 DATA DIVISION.
000050
000060 PROCEDURE DIVISION.
000070     PERFORM GREETING.
000080
000090
000100     STOP RUN.
000110
000120 GREETING.
000130
000140
000150     DISPLAY "HELLO FROM IBM.".
```

## Palabras definidas por el usuario

Dentro de un elemento de origen, una palabra definida por el usuario sólo se puede utilizar como un tipo de elemento de origen.

**Ejemplo 1**

En el ejemplo siguiente, el nombre de procedimiento es el mismo que el identificador de programa:

```
$ cat tcSameName.cbl
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. GREETING.
000030 ENVIRONMENT DIVISION.
000040 DATA DIVISION.
000050
000060 PROCEDURE DIVISION.
000070
000080 MY-PARAGRAPH.
000090     PERFORM GREETING.
000100     PERFORM END-PARAGRAPH.
000110
000120 GREETING.
000130     DISPLAY "WELCOME TO IBM COBOL para Linux 1.2.0...".
000140
000150 END-PARAGRAPH.
000160     DISPLAY "HAVE A NICE DAY!".

$ cob2 -o tcSameName tcSameName.cbl
IBM COBOL para Linux 1.2.0 compile started
0LineID  Message code  Message text
      9  IGYPS3007-S   "GREETING" was not defined as a procedure-name.
                       The statement was discarded.
-Messages    Total    Informational    Warning    Error    Severe    Terminating
0Printed:       1                                              1
End of compilation 1,  program GREETING,  highest severity: Severe.
Return code 12
```

**Ejemplo 2**

En el ejemplo siguiente, GREETING se utiliza como nombre de datos y como nombre de procedimiento:

```
$ cat tcSameName2.cbl
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. TEST-CASE.
000030 ENVIRONMENT DIVISION.
000040 DATA DIVISION.
000050
000060 WORKING-STORAGE SECTION.
000070
000080 01 GREETING PIC X(20) VALUE "FROM IBM.".
000090 PROCEDURE DIVISION.
000100 MY-PROGRAM.
000110     PERFORM GREETING.
000120     STOP RUN.
000130
000140 GREETING.
000150     DISPLAY "HELLO " GREETING.

$ cob2 -o tcSameName tcSameName2.cbl
IBM COBOL para Linux 1.2.0 compile started
0LineID  Message code  Message text
     11  IGYPS3007-S   "GREETING" was not defined as a procedure-name.
                       The statement was discarded.
-Messages    Total    Informational    Warning    Error    Severe    Terminating
0Printed:       1                                             1
End of compilation 1,  program TEST-CASE,  highest severity: Severe.
Return code 12
```

Según el estándar COBOL 2002, las palabras definidas por el usuario dentro de un elemento de origen se pueden utilizar como un solo tipo de palabra definida por el usuario.

Cambie el código fuente COBOL para que siga a COBOL 2002 estándar.

Para el Ejemplo 1:

```
$ cat sol-tcSameName.cbl
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. GREETING.
000030 ENVIRONMENT DIVISION.
000040 DATA DIVISION.
000050
000060 PROCEDURE DIVISION.
000070
000080 MY-PARAGRAPH.
000090     PERFORM HELLO.
000100     PERFORM END-PARAGRAPH.
000110
000120 HELLO.
000130     DISPLAY "WELCOME TO IBM COBOL para Linux 1.2.0...".
000140
000150 END-PARAGRAPH.
000160     DISPLAY "HAVE A NICE DAY!".
```

Para el Ejemplo 2:

```
$ cat sol-tcSameName2.cbl
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. TEST-CASE.
000030 ENVIRONMENT DIVISION.
000040 DATA DIVISION.
000050
000060 WORKING-STORAGE SECTION.
000070
000080 01 GREETING PIC X(20) VALUE "FROM IBM.".
000090 PROCEDURE DIVISION.
000100 MY-PROGRAM.
000110     PERFORM MY-GREETING.
000120     STOP RUN.
000130
000140 MY-GREETING.
000150     DISPLAY "HELLO " GREETING.
```

## SECCIÓN DE INFORME y SECCIÓN DE PANTALLA

IBM COBOL for Linux en x86 no da soporte a REPORT SECTION o SCREEN SECTION.

Cuando utiliza IBM COBOL for Linux en x86 para compilar el código fuente que contiene REPORT SECTION o SCREEN SECTION, recibirá los mensajes de error en el ejemplo siguiente:

```
$ cat tcReportScreenSection.cbl
     000010 IDENTIFICATION DIVISION.
     000020 PROGRAM-ID. TEST-CASE.
     000030 ENVIRONMENT DIVISION.
     000040 DATA DIVISION.
     000050
     000060 REPORT SECTION.
     000070 01 TYPE IS PAGE HEADING.
     000080 01 TYPE IS PAGE FOOTING.
     000090
     000100 SCREEN SECTION.

$ cob2 -o tcReportScreenSection tcReportScreenSection.cbl IBM COBOL para Linux 1.2.0 compile
started
0LineID  Message code  Message text
      6  IGYDS0148-S   "REPORT" is a reserved word related to language not supported by this
                       compiler.  The statement was discarded.
     10  IGYDS0009-E   "SCREEN" should not begin in area "A".  It was processed as if found
                       in area "B".
-Messages    Total    Informational    Warning    Error    Severe    Terminating
0Printed:       2                                   1         1
End of compilation 1,  program TEST-CASE,  highest severity: Severe.
Return code 12
```

Para obtener detalles sobre REPORT SECTION y SCREEN SECTION, consulte COBOL 2002 estándar, sección 13.7, sección Informe y sección 13.8, sección Pantalla.

## Caracteres nulos incluidos en variables

Cuando utilice IBM COBOL for Linux en x86 para compilar el código fuente que contiene una variable con caracteres nulos incorporados, se eliminarán los caracteres nulos.

Consulte la salida de MY-DATA en el ejemplo siguiente. Se eliminan los caracteres nulos.

```
$ cat tcBlankReplaceNull.cbl
     000010 IDENTIFICATION DIVISION.
     000020 PROGRAM-ID. "TEST-CASE".
     000030 ENVIRONMENT DIVISION.
     000040 CONFIGURATION SECTION.
     000050 DATA DIVISION.
     000060 WORKING-STORAGE SECTION.
     000080   77 MY-DATA PIC X(5) VALUE X"4F4E00004E".
     000120 PROCEDURE DIVISION.
     000125     DISPLAY MY-DATA "<-".
     000130     STOP RUN.
$ cob2 tcBlankReplaceNull.cbl IBM COBOL para Linux 1.2.0 compile started
End of compilation 1,  program TEST-CASE,  no statements flagged.
$ a.out
ONN<-
```

## Formato de literales de coma flotante

IBM COBOL for Linux en x86 utiliza este formato para literales de coma flotante: [+ |-] < mantissa> E [+ |-] < exponent>.

Cuando se utiliza IBM COBOL for Linux en x86 para compilar programas fuente que contienen literales de coma flotante que no utilizan una anotación electrónica; por ejemplo, si se utiliza 1500.0 en lugar de 1.5E3, recibirá los mensajes de error en el ejemplo siguiente:

```
$ cat tcFloatingFormat.cbl
     000100 IDENTIFICATION DIVISION.
     000200 PROGRAM-ID. TEST-CASE.
     000300 ENVIRONMENT DIVISION.
     000400 DATA DIVISION.
     000500
     000600 WORKING-STORAGE SECTION.
     000700 01 FLOAT1 COMP-1 VALUE 1357.9.
     000800 01 FLOAT2 COMP-1 VALUE 2468.0.
     000900
     001000 PROCEDURE DIVISION.
```

```
     001100     DISPLAY "FLOAT 1 IS " FLOAT1.
     001200     DISPLAY "FLOAT 2 IS " FLOAT2.
     001300     STOP RUN.

$ cob2 -o tcFloatingFormat tcFloatingFormat.cbl IBM COBOL para Linux 1.2.0 compile started
0LineID  Message code  Message text
      7  IGYGR1080-S   A "VALUE" clause literal was not compatible with the
                       data category of the subject data item.  The "VALUE"
                       clause was discarded.
                       Same message on line:   8
-Messages    Total    Informational    Warning    Error    Severe    Terminating
0Printed:       2                                            2
End of compilation 1,  program TEST-CASE,  highest severity: Severe.
Return code 12
```

También puede utilizar `scu` para convertir este formato de literales de coma flotante.

## Organización de archivos RECORD SEQUENTIAL

IBM COBOL for Linux en x86 no da soporte a la organización de archivos `RECORD SEQUENTIAL` .

Si el origen contiene la organización de archivos `RECORD SEQUENTIAL` , recibirá los mensajes de error en el ejemplo siguiente.

Cambie `RECORD SEQUENTIAL` por `LINE SEQUENTIAL`y, a continuación, vuelva a compilar. También puede utilizar `scu` para sustituir `RECORD SEQUENTIAL` por `LINE SEQUENTIAL`.

```
$cat tcRecordSequential.cbl
     000010 IDENTIFICATION DIVISION.
     000020 PROGRAM-ID. TEST-CASE.
     000030 ENVIRONMENT DIVISION.
     000031
     000032 INPUT-OUTPUT SECTION.
     000033 FILE-CONTROL.
     000034     SELECT CONTACTS
     000035        ASSIGN TO "MYFILE.DAT"
     000036        ORGANIZATION IS RECORD SEQUENTIAL.
     000037
     000040 DATA DIVISION.
     000050 FILE SECTION.
     000051 FD CONTACTS
     000052     RECORD CONTAINS 20 CHARACTERS.
     000053 01 CONTACT-RECORD.
     000054    05 FNAME  PIC X(10).
     000055    05 LNAME  PIC X(10).
     000073
     000090 PROCEDURE DIVISION.
     000100     PERFORM OPEN-FILE.
     000110     PERFORM INPUT-DATA.
     000120     PERFORM SAVE-DATA.
     000130     PERFORM CLOSE-FILE.
     000140     PERFORM END-PROGRAM.
     000150
     000160 INPUT-DATA.
     000170     DISPLAY "ENTER FIRST NAME.".
     000180     ACCEPT FNAME.
     000190     DISPLAY "ENTER LAST NAME.".
     000200     ACCEPT LNAME.
     000210
     000220 OPEN-FILE.
     000230     OPEN OUTPUT CONTACTS.
     000240
     000250 SAVE-DATA.
     000260     WRITE CONTACT-RECORD.
     000270
     000280 CLOSE-FILE.
     000290     CLOSE CONTACTS.
     000300
     000310 END-PROGRAM.
     000320     STOP RUN.
$ cob2 -o tcRecordSequential tcRecordSequential.cbl IBM COBOL para Linux 1.2.0 compile started
0LineID  Message code  Message text
      9  IGYLN2929-S   Incorrect ORGANIZATION type detected at RECORD.  Clause ignored.
      9  IGYLN1357-S   Expected a data-name; SEQUENTIAL found. Statement or clause ignored.
     13  IGYGR1216-I   A "RECORDING MODE" of "F" was assumed for file "CONTACTS".
-Messages    Total    Informational    Warning    Error    Severe    Terminating
0Printed:       3           1                                 2
```

```
End of compilation 1,  program TEST-CASE,  highest severity: Severe.
Return code 12
```

# Representación de datos

La representación de los datos puede diferir entre los compiladores noIBM y COBOL for Linux en x86.

## Datos binarios

Los compiladores IBM COBOL utilizan la representación nativa de la plataforma al manejar elementos de datos binarios (BINARY, COMP, COMP-4 y COMP-5). En Linux x86, los elementos de datos binarios se almacenarán y manipularán en formato little-endian (dígito menos significativo en la dirección más baja).

Al migrar aplicaciones COBOL desde compiladores noIBM a COBOL for Linux en x86, es posible que obtenga resultados inesperados si el programa accede a datos almacenados en formato big-endian, ya que el compilador y el tiempo de ejecución tratarán los datos como formato little-endian de forma predeterminada.

Puede utilizar la opción `BINARY(BE)` para informar al compilador COBOL for Linux en x86 que maneje elementos de datos BINARY, COMP y COMP-4 en formato big-endian coherentes con compiladores noIBM que representan elementos binarios como big-endian. Sin embargo, esto tendrá cierta sobrecarga de rendimiento, ya que el compilador necesita convertir los datos a y desde su formato nativo, little-endian. Los elementos de datos COMP-5 no se ven afectados por la opción `BINARY(BE)` o `BINARY(LE)` ya que COMP-5 indica un elemento de datos binario nativo que utiliza la representación nativa de la plataforma. Si utiliza una combinación de COMP-5 y otros tipos de datos BINARY/COMP/COMP-4 en el programa, tenga cuidado al utilizar la opción `BINARY(BE)` . Si un elemento de datos determinado necesita permanecer en representación little-endian (LE) cuando se ha especificado `BINARY(BE)` , utilice la cláusula `NATIVE` en la sentencia USAGE .

**Nota:**

- Micro Focus Visual Studio y COBOL-IT almacenan datos BINARY/COMP/COMP-4 en formato big endian, independientemente de la plataforma, y datos COMP-5 en el entorno nativo de la plataforma, por lo que al utilizar COBOL for Linux en x86, es posible que tenga que utilizar la opción `BINARY(BE)` para trabajar con datos de forma compatible con dichos compiladores.
- Si está utilizando IBM MQ, se espera que los parámetros de API estén en formato big endian, por lo que deberá utilizar la opción BINARY (BE) y FLOAT (BE) al trabajar con MQ en Linux.
- IBM Db2y Oracle Pro *COBOL añaden sus propias áreas de datos en el programa COBOL generado para comunicarse con sus bibliotecas de cliente. Estas bibliotecas de cliente esperan datos en formato little endian nativo en Linux, incluso si dan soporte a datos de host big-endian. Al trabajar con Db2 o Pro *COBOL, debe utilizar el formato binario nativo predeterminado (BINARY (NATIVE) o BINARY (LE)).
- Puesto que IBM MQ espera un formato binario diferente que IBM Db2 y Oracle Pro *COBOL, no se recomienda tener llamadas MQ y SQL en la misma unidad de compilación o compilación por lotes.

## Datos nacionales

Los compiladores IBM COBOL utilizan la representación nativa UTF-16 de la plataforma al manejar datos nacionales. En Linux x86, los elementos de datos nacionales se almacenarán y manipularán en formato little endian UTF-16 .

Al migrar aplicaciones COBOL desde compiladores noIBM a COBOL for Linux en x86, es posible que obtenga resultados inesperados si el programa accede a datos almacenados en formato big-endian, ya que el compilador y el tiempo de ejecución tratarán los datos como formato little-endian de forma predeterminada.

Puede utilizar la opción `UTF16(BE)` para informar al compilador COBOL for Linux en x86 para manejar elementos de datos nacionales en formato big-endian coherentes con compiladores que no son deIBM que representan elementos de datos nacionales como big-endian. Sin embargo, esto tendrá cierta sobrecarga de rendimiento, ya que el compilador necesita convertir los datos a y desde su formato nativo,

little-endian. Si un elemento de datos determinado necesita permanecer en representación little-endian (LE) cuando se ha especificado UTF16(BE), utilice la cláusula NATIVE en la sentencia USAGE.

**Tareas relacionadas**
"Establecimiento del entorno local" en *Guía de programación*
"Arreglo de diferencias causadas por representaciones de datos" en la publicación *Programming Guide*

**Referencias relacionadas**
"CHAR" en la *Guía de programación*
"SOSI" en la *Guía de programación*

## Cambios de DIVISIÓN DE IDENTIFICACIÓN

IDENTIFICATION DIVISION debe ser la primera división en cada programa fuente COBOL.

Cuando se utiliza IBM COBOL for Linux en x86 para compilar el código fuente que contiene un IDENTIFICATION DIVISION que no es la primera división en un programa fuente COBOL, recibirá los mensajes de error en el ejemplo siguiente:

```
$ cat tcIdentificationDivision.cbl
000010 ENVIRONMENT DIVISION.
000020 DATA DIVISION.
000030
000040 WORKING-STORAGE SECTION.
000050 01 GREETING PIC X(50) VALUE "HELLO FROM IBM...".
000060 01 GOODBYE PIC X(50) VALUE "HAVE A NICE DAY...".
000070
000080 IDENTIFICATION DIVISION.
000090 PROGRAM-ID. TEST-CASE.
000100
000110 PROCEDURE DIVISION.
000120     DISPLAY GREETING.
000130     DISPLAY GOODBYE.
000140     STOP RUN.

$ cob2 -o tcIdentificationDivision tcIdentificationDivision.cbl
IBM COBOL para Linux 1.2.0 compile started
0LineID  Message code  Message text
      1  IGYLN0034-E   Program-name expected; Default program-name COBOLPGM00 assumed.
     12  IGYPS2121-S   "GREETING" was not defined as a data-name.  The statement was
                       discarded.
                       Same message on line:     12
     13  IGYPS2121-S   "GOODBYE" was not defined as a data-name.  The statement was
                       discarded.
                       Same message on line:     13
     14  IGYSC0136-E   End of source file was encountered or an "END PROGRAM" marker for a
                       containing program was found, but an "END PROGRAM" marker for program
                       "TEST-CASE" was not found.  "END PROGRAM TEST-CASE" was assumed.
     14  IGYSC0136-E   End of source file was encountered or an "END PROGRAM" marker for a
                       containing program was found, but an "END PROGRAM" marker for program
                       "COBOLPGM00" was not found.  "END PROGRAM COBOLPGM00" was assumed.
-Messages    Total    Informational    Warning    Error    Severe    Terminating
0Printed:       7                                  3         4
End of compilation 1,  program COBOLPGM00,  highest severity: Severe.
Return code 12
```

Según COBOL 2002 estándar, una unidad de origen es un conjunto de sentencias COBOL tal como se especifica en este documento y consta de un IDENTIFICATION DIVISION seguido opcionalmente por cualquiera de las secciones ENVIRONMENT DIVISION, DATA DIVISIONo PROCEDURE DIVISION, o una combinación de estas divisiones.

Cambie el código fuente COBOL para que siga COBOL 2002 estándar:

```
$ cat sol-tcIdentificationDivision.cbl
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. TEST-CASE.
000030 ENVIRONMENT DIVISION.
000040 DATA DIVISION.
000050
000060 WORKING-STORAGE SECTION.
000070 01 GREETING PIC X(50) VALUE "HELLO FROM IBM...".
000080 01 GOODBYE PIC X(50) VALUE "HAVE A NICE DAY...".
000090
```

```
000100 PROCEDURE DIVISION.
000110     DISPLAY GREETING.
000120     DISPLAY GOODBYE.
000130     STOP RUN.
```

# Cambios de DATA DIVISION

Para ejecutar los programas fuente en IBM COBOL for Linux en x86, realice los cambios necesarios de `DATA DIVISION` en los programas fuente.

## Cláusulas OCCURS en el nivel 01

Según COBOL 2002 estándar, debe seguir las reglas de sintaxis para la cláusula OCCURS .

Cuando se utiliza IBM COBOL for Linux en x86 para compilar el código fuente que contiene una cláusula OCCURS en el nivel 01, recibirá los mensajes de error en el ejemplo siguiente:

```
$ cat tcOccursClause.cbl
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. TEST-CASE.
000030 ENVIRONMENT DIVISION.
000040 DATA DIVISION.
000050
000060 WORKING-STORAGE SECTION.
000061
000062
000063 01 EMPLOYEES OCCURS 100 TIMES.
000064    05 FNAME PIC X(10).
000065    05 LNAME PIC X(10).
000066    05 SALARY PIC 9(5).
000067
000068 77 COUNTER PIC 9.
000070
000071 PROCEDURE DIVISION.
000072     MOVE "DAVID" TO FNAME(1).
000073     MOVE "SMITH" TO LNAME(1).
000074     MOVE "60000" TO SALARY(1).
000075     MOVE "JENNY" TO FNAME(2).
000076     MOVE "HU" TO LNAME(2).
000077     MOVE "55000" TO SALARY(2).
000078
000080 DISPLAY-RESULT.
000081     PERFORM VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 2
000082      DISPLAY "EMPLOYEE: " FNAME IN EMPLOYEES(COUNTER)" "
000083               LNAME IN EMPLOYEES(COUNTER)" "
000084              "Salary: " SALARY IN EMPLOYEES(COUNTER)
000085     END-PERFORM.
000086 STOP-PROGRAM.
000090     STOP RUN.
$ cob2 -o tcOccursClause tcOccursClause.cbl
IBM COBOL for Linux 1.2.0 compile started
0LineID  Message code  Message text
      9  IGYDS1063-E   An "OCCURS" clause was found in the definition of a level-1 item.  The
                       "OCCURS" clause was discarded.
-Messages    Total    Informational    Warning    Error    Severe    Terminating
0Printed:        1                                  1
End of compilation 1,  program TEST-CASE,  highest severity: Error.
Return code 8
```

Según COBOL 2002 estándar, la cláusula OCCURS no se puede especificar en una entrada de descripción de datos que tenga cualquiera de los elementos siguientes:

- Un número de nivel de 01, 66, 77 o 88
- Un elemento de datos de aparición de variable subordinado al mismo

Cambie el código fuente COBOL para que siga a COBOL 2002 estándar. Por ejemplo, si cambia el código del ejemplo anterior utilizando la definición de EMPLOYEES que se mueve del nivel 01 al nivel 03, se compila sin mensajes.

```
$ cat sol-tcOccursClause.cbl
     000010 IDENTIFICATION DIVISION.
     000020 PROGRAM-ID. TEST-CASE.
     000030 ENVIRONMENT DIVISION.
```

```
     000040 DATA DIVISION.
     000050
     000060 WORKING-STORAGE SECTION.
     000061
     000062 01.
     000063 03 EMPLOYEES OCCURS 100 TIMES.
     000065    05 FNAME PIC X(10).
     000066    05 LNAME PIC X(10).
     000067    05 SALARY PIC 9(5).
     000068
     000069 77 COUNTER PIC 9.000070
     000071 PROCEDURE DIVISION.
     000072     MOVE "DAVID" TO FNAME(1).
     000073     MOVE "SMITH" TO LNAME(1).
     000074     MOVE "60000" TO SALARY(1).
     000075     MOVE "JENNY" TO FNAME(2).
     000076     MOVE "HU"    TO LNAME(2).
     000077     MOVE "55000" TO SALARY(2).
     000078
     000080 DISPLAY-RESULT.
     000082      PERFORM VARYING COUNTER FROM 1 BY 1 UNTIL COUNTER > 2
     000084       DISPLAY "EMPLOYEE: " FNAME IN EMPLOYEE(COUNTER) " "
     000085               LNAME IN EMPLOYEE(COUNTER) " "
     000085              "Salary: " SALARY IN EMPLOYEE(COUNTER)
     000086      END-PERFORM.
     000087 STOP-PROGRAM.
     000120      STOP RUN.
```

## COMP-X

IBM COBOL for Linux en x86 no da soporte a COMP-X.

Cuando se utiliza IBM COBOL for Linux en x86 para compilar el código fuente que contiene COMP-X, recibirá el mensaje de error en el ejemplo siguiente:

```
$ cat tcCompX.cbl
     000010 IDENTIFICATION DIVISION.
     000020 PROGRAM-ID. TEST-CASE.
     000030 ENVIRONMENT DIVISION.
     000040 DATA DIVISION.
     000050
     000060 WORKING-STORAGE SECTION.
     000070 01 STU-ID PIC S9(4) COMP-X VALUE ZERO.

$ cob2 -o tcCompX tcCompX.cbl
IBM COBOL for Linux 1.2.0 compile started
0LineID  Message code  Message text
      7  IGYDS1089-S   "COMP-X" was invalid.  Scanning was resumed at the
                       next area "A" item, level-number, or the start of
                       the next clause
-Messages    Total    Informational    Warning    Error    Severe    Terminating
0Printed:       1                                          1
End of compilation 1,  program TEST-CASE,  highest severity: Severe.
Return code 12
```

Cambie COMP-X por COMP-5y realice los cambios correspondientes en la cláusula PICTURE . Debe conservar el mismo almacenamiento asignado para el elemento de datos.

La tabla siguiente muestra la asignación de almacenamiento de COMP-5 :

*Tabla 1. Asignación de almacenamiento COMP-5*

| **Imagen** | **Representación de almacenamiento** |
|---|---|
| S9(1) a S9(4) | Media palabra binaria (2 bytes) |
| S9(5) a S9(9) | Palabra completa binaria (4 bytes) |
| S9(10) a S9(18) | Palabra doble binaria (8 bytes) |
| 9(1) a 9(4) | Media palabra binaria (2 bytes) |
| 9(5) a 9(9) | Palabra completa binaria (4 bytes) |
| 9(10) a 9(18) | Palabra doble binaria (8 bytes) |

## Errores de tiempo de ejecución debido a variables no inicializadas que se están convirtiendo a un tipo de visualización

Recibirá mensajes de error de tiempo de ejecución si el remitente de una sentencia se mueve al destinatario pero el remitente no contiene datos válidos para el tipo de destinatario, posiblemente porque la variable que se está moviendo no se ha inicializado.

Al ejecutar un programa que contiene una variable no inicializada, recibirá los mensajes de error en el ejemplo siguiente:

```
$cat tcConvert2DisplayType.cbl
     000010 IDENTIFICATION DIVISION.
     000020 PROGRAM-ID. "TEST-CASE".
     000030 ENVIRONMENT DIVISION.
     000040 CONFIGURATION SECTION.
     000050 DATA DIVISION.
     000060 WORKING-STORAGE SECTION.
     000070 01 EMP-INFO.
     000080    05  EMP-ID PIC S9(5) sign is trailing separate.
     000090 01 EMP-ID2 PIC S9(4).
     000100 PROCEDURE DIVISION.
     000120     MOVE EMP-ID OF EMP-INFO TO EMP-ID2.
     000130     STOP RUN.
```

```
$ cob2 -o tcConvert2DisplayType tcConvert2DisplayType.cbl
IBM COBOL for Linux 1.2.0 compile started
End of compilation 1,  program TEST-CASE,  no statements flagged.
> tcConvert2DisplayType
  Traceback:
/opt/ibm/cobol/rte/usr/lib/libcob2_64r.so(+0xa6302)[0x7f46046d9302]
/opt/ibm/cobol/rte/usr/lib/libcob2_64r.so(writeERRmsg+0x72e)[0x7f46046b0d2e]
/opt/ibm/cobol/rte/usr/lib/libcob2_64r.so(+0x7e762)[0x7f46046b1762]
/opt/ibm/cobol/rte/usr/lib/libcob2_64r.so(_iwzcBCD_CONV_ZndTS_To_ZndTO+0x356)[0x7f460466f446]
tcConvert2DisplayType(TEST-CASE+0xce)[0x560b45e13eca]
    --- End of call chain ---
IWZ040S  An invalid separate sign was detected.
IWZ901S  Program exits due to severe or critical error.

Aborted (core dumped)
```

Inicialice EMP-ID de EMP-INFO en 0:

```
$ cat sol-tcConvert2DisplayType.cbl
     000010 IDENTIFICATION DIVISION.
     000020 PROGRAM-ID. "TEST-CASE".
     000030 ENVIRONMENT DIVISION.
     000040 CONFIGURATION SECTION.
     000050 DATA DIVISION.
     000060 WORKING-STORAGE SECTION.
     000070 01 EMP-INFO.
     000080    05  EMP-ID PIC S9(5) SIGN IS TRAILING SEPARATE
                VALUE ZERO.
     000090 01 EMP-ID2 PIC S9(4).
     000100 PROCEDURE DIVISION.
     000120     MOVE EMP-ID OF EMP-INFO TO EMP-ID2.
     000130     STOP RUN.
```

Además de inicializar EMP-ID en 0, puede compilar utilizando -qwsclear, que borra WORKING-STORAGE de un programa a ceros binarios cuando se inicializa el programa. El almacenamiento se borra antes de que se apliquen las cláusulas VALUE . Sin embargo, este enfoque no es el preferido debido a consideraciones de rendimiento.

## Elementos de datos redefinidos y cláusulas OCCURS

IBM COBOL for Linux en x86 no da soporte a la redefinición de un elemento de datos que tenga una cláusula OCCURS.

Cuando se utiliza IBM COBOL for Linux en x86 para compilar el código fuente donde se redefine un elemento de datos con la cláusula OCCURS, recibirá los mensajes de error en el ejemplo siguiente:

```
$ cat tcRedefineOccurs.cbl
     01 EMPLOYEE-INFO.
        03 EMPLOYEE OCCURS 50 TIMES.
              05 NAME PIC X(30).
              05 AGE PIC 9(3).
        03 NEW-EMPLOYEE REDEFINES EMPLOYEE.
           04 G2 OCCURS 50.
              05 FNAME PIC X(10).
              05 LNAME PIC X(20).
              05 AGE PIC 9(3).

$ cob2 -o tcRedefineOccurs tcRedefineOccurs.cbl
IBM COBOL para Linux 1.2.0 compile started
0LineID Message code Message text
     10 IGYLN1222-S EMPLOYEE contains a OCCURS clause. Clause ignored.
Messages Total Informational Warning Error Severe Terminating
Printed: 1 1
End of compilation 1, program PGM1, highest severity: Severe.
Return code 12
```

Según COBOL 2002 estándar, el elemento de datos que se está redefiniendo no puede contener una cláusula OCCURS .

Cambie el código fuente COBOL para que siga a COBOL 2002 estándar. Por ejemplo, si cambia el código en el ejemplo anterior añadiendo un elemento de grupo y renumerando, se compila sin mensajes.

```
$ cat tcRedefineOccurs.cbl
     01 EMPLOYEE-INFO.
        02 G1.
           03 EMPLOYEE OCCURS 50 TIMES.
              05 NAME PIC X(30).
              05 AGE PIC 9(3).
        02 G2 REDEFINES G1.
           03 EMPLOYEE OCCURS 50 TIMES.
              05 FNAME PIC X(10).
              05 LNAME PIC X(20).
              05 AGE PIC 9(3).
```

# Cambios de PROCEDURE DIVISION

Para ejecutar los programas fuente en IBM COBOL for Linux en x86, realice los cambios necesarios de `PROCEDURE DIVISION` en los programas fuente.

## Mover elementos de datos nacionales a elementos de datos alfanuméricos

No puede mover un elemento de datos nacional a un elemento de datos alfanumérico.

Cuando se utiliza IBM COBOL for Linux en x86 para compilar el código fuente que contiene una sentencia `MOVE` de un elemento de datos nacional a un elemento de datos alfanumérico, recibirá el mensaje de error en el ejemplo siguiente:

```
$ cat tcNational2Alphanumeric.cbl
     000010 IDENTIFICATION DIVISION.
     000020 PROGRAM-ID. ND2AD.
     000030 ENVIRONMENT DIVISION.
     000040 CONFIGURATION SECTION.
     000050
     000060 DATA DIVISION.
     000070 WORKING-STORAGE SECTION.
     000080
     000090 01 DataNational      PIC N(100).
     000100
     000110 01 DataAlphanumeric    PIC X(100).
     000120
     000130 PROCEDURE DIVISION.
     000140       MOVE DataNational TO DataAlphanumeric.
     000150 STOP-MY-PROGRAM.
     000160       STOP RUN.
```

```
$ cob2 -o tcNational2Alphanumeric tcNational2Alphanumeric.cbl IBM COBOL for Linux 1.2.0
compile started 0LineID  Message code  Message text      14  IGYPA3005-S   "DATANATIONAL
(NATIONAL ITEM)" and "DATAALPHANUMERIC (ALPHANUMERIC)"                          did not
follow the "MOVE" statement compatibility rules.  The
statement was discarded.                                              -Messages
Total    Informational    Warning    Error    Severe    Terminating 0Printed:
1                                               1            End of compilation 1,  program
ND2AD,  highest severity: Severe. Return code 12
```

Según COBOL 2002 estándar, no puede mover un elemento de datos nacional a un elemento de datos alfanumérico.

Cambie el código fuente COBOL para que siga a COBOL 2002 estándar.

## Errores de símbolo no definidos para las funciones C que se están llamando

Utilice la opción de compilador PGMNAME para controlar cómo maneja el compilador los nombres de programa.

Cuando se utiliza IBM COBOL for Linux en x86 para compilar el código fuente que contiene una llamada a una función C en caracteres mixtos o en minúsculas, recibirá los mensajes de error en el ejemplo siguiente:

```
$ cat cTest.c
     void CFunction(){
     printf("HELLO FROM C...\n");
     }

$ cat tcCallCFunction.cbl
     000010 IDENTIFICATION DIVISION.
     000020 PROGRAM-ID. TEST-CASE.
     000030 ENVIRONMENT DIVISION.
     000040 CONFIGURATION SECTION.
     000050 DATA DIVISION.
     000060 WORKING-STORAGE SECTION.
     000070
     000080 PROCEDURE DIVISION.
     000090     CALL "CFunction"
     000100     DISPLAY "HELLO FROM COBOL...".
     000110     STOP RUN.

$ gcc -m64 -fPIC -c cTest.c  $ cob2 -o tcCallCFunction tcCallCFunction.cbl cTest.o IBM
COBOL for Linux 1.2.0 compile started End of compilation 1,  program TEST-CASE,  no
statements flagged. tcCallCFunction.o: In function `TEST-CASE': /home/user1/tcCallCFunction.cbl:
(.text+0xb5): undefined reference to `CFUNCTION' tcCallCFunction.o:(.rodata+0x50): undefined
reference to `CFUNCTION' collect2: error: ld returned 1 exit status
```

IBM COBOL for Linux en x86 convierte los nombres de programa en mayúsculas. Si tiene código fuente COBOL que contiene una llamada a una función C en caracteres mixtos o en minúsculas, esta función se convierte en caracteres en mayúsculas. El enlazador no encontrará el programa y se visualiza un mensaje de error para indicar un símbolo no resuelto.

Puede utilizar la opción de compilador PGMNAME para controlar cómo maneja el compilador los nombres de programa. El valor predeterminado es PGMNAME(UPPER), pero puede utilizar PGMNAME(MIXED) para procesar el nombre de programa tal cual, sin truncamiento, conversión o conversión a mayúsculas. Cuando especifique PGMNAME(MIXED), utilice el formato literal del nombre de programa, es decir, haga que el nombre de programa sea una serie literal como "programname", o verá el mensaje siguiente:

```
IGYDS1046-E   A user-defined word was found as a "PROGRAM-ID" name under
the "PGMNAME(LONGMIXED)" compiler option.
```

Para ver un ejemplo sobre un programa COBOL que llama a funciones C, consulte *Ejemplo: Programa COBOL que llama a funciones C* en COBOL for Linux en x86 Guía de programación.

```
$ cat sol-tcCallCFunction.cbl
000010 CBL PGMNAME(LONGMIXED)
000020 IDENTIFICATION DIVISION.
000030 PROGRAM-ID. "TEST-CASE".
000040 ENVIRONMENT DIVISION.
000050 CONFIGURATION SECTION.
000060 DATA DIVISION.
```

```
000070 WORKING-STORAGE SECTION.
000080
000090 PROCEDURE DIVISION.
000100     CALL "CFunction"
000110     DISPLAY "HELLO FROM COBOL...".
000120     STOP RUN.
$ cob2 -o sol-tcCallCFunction sol-tcCallCFunction.cbl cTest.o
IBM COBOL for Linux 1.2.0 compile started
End of compilation 1,  program TEST-CASE,  no statements flagged.
$ sol-tcCallCFunction
HELLO FROM C...
HELLO FROM COBOL...
```

## Programas COBOL que contienen una sentencia ACCEPT

En IBM COBOL for Linux en x86, la sentencia ACCEPT asigna una 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.
- Si la línea de entrada es más larga que el elemento de datos:
  - Cuando un programa lee desde una pantalla, los caracteres restantes se descartan.
  - Cuando un programa lee de un archivo, los caracteres restantes se conservan como la siguiente línea de entrada para el archivo.

Si los datos de entrada son más largos que el área de recepción, IBM COBOL for Linux en x86 rellena el área con espacios de la representación adecuada para el área de recepción.

Según COBOL 2002 estándar, el implementador debe definir el tamaño de una transferencia de datos para cada dispositivo de hardware.

## Datos con formato BINARY en una sentencia ACCEPT

Según COBOL 2002 estándar, la sentencia ACCEPT provoca la transferencia de datos desde el dispositivo de hardware. Estos datos sustituyen el contenido del elemento de datos al que hace referencia *identifier-1*. El implementador define cualquier conversión de datos necesaria entre el dispositivo de hardware y el elemento de datos al que hace referencia *identifier-1* . La implementación de IBM COBOL for Linux en x86 solo permite datos visualizables (visualización, nacional, numérico, alfanumérico) que excluyen los datos con formato binario.

Cuando se utiliza IBM COBOL for Linux en x86 para compilar código fuente que contiene datos con formato BINARY en una sentencia ACCEPT , recibirá el mensaje de error en el ejemplo siguiente:

```
$ cat tcAcceptBinary.cbl
     000010 IDENTIFICATION DIVISION.
     000020 PROGRAM-ID. TEST-CASE.
     000030 ENVIRONMENT DIVISION.
     000040 DATA DIVISION.
     000050
     000060 WORKING-STORAGE SECTION.
     000070 01 EMPLOYEE-ID PIC S9(4) COMP-5 VALUE ZERO.
     000080
     000090 PROCEDURE DIVISION.
     000100 MY-PROGRAM.
     000110     DISPLAY "PLEASE INSERT EMPLOYEE'S ID NUMBER..".
     000120     ACCEPT EMPLOYEE-ID.
     000130
     000140     STOP RUN.

$ cob2 -o tcAcceptBinary tcAcceptBinary.cbl IBM COBOL for Linux 1.2.0 compile
started 0LineID  Message code  Message text      12  IGYLN0642-E   Statement
accepted, but ACCEPT identifier may not achieve expected
conversion.                                                    -Messages    Total
Informational    Warning    Error    Severe    Terminating 0Printed:
1                           1                        End of compilation 1,  program
TEST-CASE,  highest severity: Error. Return code 8
```

Cambie el tipo de datos de COMP-5 a DISPLAYy realice también más cambios en el origen de entrada.

```
$ cat sol-tcAcceptBinary.cbl
     000010 IDENTIFICATION DIVISION.
     000020 PROGRAM-ID. TEST-CASE.
     000030 ENVIRONMENT DIVISION.
     000040 DATA DIVISION.
     000050
     000060 WORKING-STORAGE SECTION.
     000070 01 EMP-ID PIC 9(4) VALUE ZEROES.
     000080 01 EMPLOYEE-ID PIC S9(4) COMP-5 VALUE ZERO.
     000090
     000100 PROCEDURE DIVISION.
     000110 MY-PROGRAM.
     000120     DISPLAY "PLEASE INSERT EMPLOYEE'S ID NUMBER..".
     000130     ACCEPT EMP-ID.
     000140     MOVE EMP-ID TO EMPLOYEE-ID.
     000150     STOP RUN.
```

## Parámetros NULL en sentencias CALL

IBM COBOL for Linux en x86 no da soporte a los parámetros NULL en una sentencia CALL .

Cuando se utiliza IBM COBOL for Linux en x86 para compilar código fuente que contiene parámetros NULL en una sentencia CALL , recibirá el mensaje de error en el ejemplo siguiente:

```
$ cat tcCallNull.cbl
     000010 IDENTIFICATION DIVISION.
     000020 PROGRAM-ID. TEST-CASE.
     000030 ENVIRONMENT DIVISION.
     000040 DATA DIVISION.
     000050
     000060 WORKING-STORAGE SECTION.
     000070
     000080 PROCEDURE DIVISION.
     000090     CALL "TEST-CASE2" USING NULL.
     000100     STOP RUN.

$ cob2 -o tcCallNull tcCallNull.cbl IBM COBOL for Linux 1.2.0 compile
started 0LineID  Message code  Message text       9  IGYPS2000-S   Expected a
data-name, but found "NULL".  The "CALL" statement was
discarded.                                                    -Messages    Total
Informational    Warning    Error    Severe    Terminating 0Printed:
1                                             1            End of compilation 1,  program
TEST-CASE,  highest severity: Severe. Return code 12
```

Según COBOL 2002 estándar, las reglas de sintaxis no identifican NULL como un parámetro válido para la sentencia CALL .

```
$ cat sol-tcCallNull.cbl
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. TEST-CASE.
000030 ENVIRONMENT DIVISION.
000040 DATA DIVISION.
000050
000060 WORKING-STORAGE SECTION.
000070 01 NULLPTR USAGE POINTER VALUE NULL.
000080
000110 PROCEDURE DIVISION.
000140     CALL "TEST-CASE2" USING NULLPTR.
000150     STOP RUN.
```

# Salida para números positivos

IBM COBOL for Linux en x86 no contiene un signo más (+) para los números positivos.

Por ejemplo, cuando utiliza IBM COBOL for Linux en x86 para compilar el código fuente que contiene un número positivo, la salida no muestra el signo más (+):

```
$ cat tcPlusSign.cbl
     000010 IDENTIFICATION DIVISION.
     000020 PROGRAM-ID. TEST-CASE.
     000030 ENVIRONMENT DIVISION.
```

```
      000040
      000050 INPUT-OUTPUT SECTION.
      000060
      000070 DATA DIVISION.
      000080 WORKING-STORAGE SECTION.
      000090 01 POSITIVE-NUM PIC S9(4) COMP-5 VALUE ZERO.
      000100 01 NEGATIVE-NUM PIC S9(4) COMP-5 VALUE ZERO.
      000110
      000120 PROCEDURE DIVISION.
      000130     COMPUTE POSITIVE-NUM = 10 - 3.
      000140     COMPUTE NEGATIVE-NUM = 3 - 10.
      000150     DISPLAY "10 - 3 = " POSITIVE-NUM.
      000160     DISPLAY "3 - 10 = " NEGATIVE-NUM.
      000170     STOP RUN.

$ cob2 -o tcPlusSign tcPlusSign.cbl IBM COBOL for Linux 1.2.0 compile started End of
compilation 1,  program TEST-CASE,  no statements flagged.

$ tcPlusSign2
10 - 3 = +0007
3 - 10 = -0007
```

**Nota:** Esta regla no se aplica a los elementos editados numérica-editados. Si utiliza un símbolo de control de signo de edición en una variable declarada con USAGE DISPLAY o NATIONAL, como el signo más (+) en el ejemplo siguiente:

```
000090 01 POSITIVE-NUM PIC +9(4).
000100 01 NEGATIVE-NUM PIC +9(4).
```

La salida es la siguiente:

```
10 - 3 = +0007
3 - 10 = -0007
```

## Programa de utilidad de conversión de origen (scu)

El programa de utilidad de conversión de origen, scu, es un programa autónomo Linux que ayuda a la conversión de programas fuente COBOL desde formatos de origen que no sonIBM o de formato libre a un formato que puede ser compilado por COBOL for Linux en x86.

Scu realiza las transformaciones siguientes:

- Convierte los caracteres de espacio en blanco en espacios verdaderos, incluida la expansión de tabuladores, con controles opcionales
- Normaliza los caracteres de fin de línea
- Reposiciona elementos para empezar en las áreas adecuadas, como el área de indicador, el área A o el área B, según sea necesario
- Garantiza una alineación especial para las sentencias CBL (PROCESS)
- Opcionalmente, deja en blanco los números de secuencia en las columnas 1 a 6 y elimina los números de serie en las columnas 73 a 80
- Convierte la entrada anómala de formato de origen fijo de forma predeterminada y, opcionalmente, convierte la entrada de formato libre

También realiza los siguientes arreglos semánticos y de sintaxis:

- Añade espacios perdidos alrededor de literales entrecomillados
- Ajusta la cláusula OCCURS al nivel 02 si está en el nivel 01
- Convierte literales de coma no flotante en notaciones de constante flotante
- Arregla periodos adicionales y fuera de lugar
- Convierte sentencias SET en MOVE cuando lo requieren los tipos de datos
- Convierte "< >" en "NOT ="
- Sustituye VALUES por VALUE
- Mueve el nivel 01 al área A

- Sustituye RECORD SEQUENTIAL por LINE SEQUENTIAL

**Restricciones:**

- Scu convierte las sentencias SET a MOVE sólo para los siguientes tipos de datos:
  - COMP, COMP-4o BINARY
  - COMP-3 o PACKED-DECIMAL
  - COMP-5
  - DISPLAY
  - NATIONAL (no literal)
- Scu puede convertir incorrectamente la sentencia SET a MOVE si la sentencia abarca dos o más líneas.
- Scu sustituye VALUES por VALUE sólo cuando VALUES va seguido de un literal en la misma línea.
- Scu convierte literales de coma no flotante en notaciones constantes flotantes sólo cuando los literales y VALUE (o VALUES) están en la misma línea.
- Scu ignora la sentencia REPLACE y la frase REPLACE de la sentencia COPY.

**Consejos:**

- Para obtener más información sobre el área A y el área B, consulte "Formato de referencia" en *Consulta de lenguaje*.
- Normalmente, scu realiza cambios de transformación iniciales y, a continuación, corrige errores de sintaxis y semánticos (la opción -N y los libros de copias son excepciones). Sin embargo, si scu detecta problemas graves durante la transformación, como por ejemplo una línea especial estándar no COBOL, scu generará mensajes de error y omitirá la sintaxis y la fase semántica. La opción -N y las excepciones del libro de copias:
  - Cuando se especifica -N , scu sólo realiza los cambios de transformación iniciales.
  - Los errores de sintaxis y semánticos en los libros de copias se arreglan cuando estos errores se arreglan para el archivo de origen principal con la opción -G especificada. Para obtener más información sobre cómo procesar libros de copias con scu, consulte "Opciones del programa de utilidad de conversión de origen (scu)" en *Consulta de lenguaje*.

El archivo de salida contiene código COBOL compatible que incluye todos los arreglos que scu puede proporcionar. Si no especifica la opción -o para un nombre de archivo de salida, el archivo de salida predeterminado es *filename*.scu.cbl, donde *filename* es el nombre de archivo original sin el sufijo. Por ejemplo, el nombre de archivo de salida para el origen abc.def.cob es abc.def.scu.cbl.

scu emite códigos de retorno que indican un éxito o un fracaso de la conversión de programa. Consulte los códigos de retorno y las explicaciones correspondientes en la tabla siguiente:

*Tabla 2. Códigos de retorno de Scu*

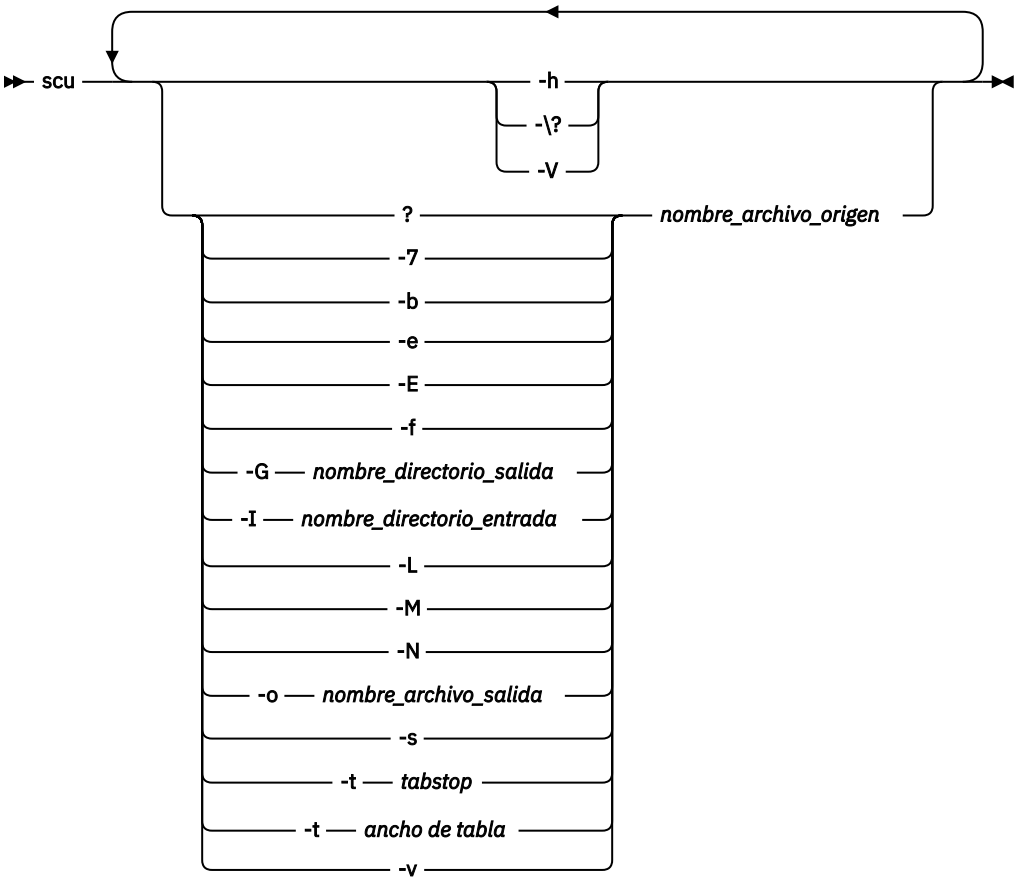
| **Código de retorno** | **Explicación** |
|---|---|
| 0 | Para indicar que scu se ejecuta correctamente. |
| 1 - 4 | Reservado para uso futuro para indicar éxito. |
| 5 | Para indicar una anomalía general. |
| 6 | Indica que no se ha podido abrir un archivo de copia. |

**Atención:** Scu convierte los programas fuente a un formato que se puede compilar con COBOL for Linux, pero es posible que scu no identifique o arregle todas las incompatibilidades que existen en el código. IBM no garantiza que los cambios realizados por scu estén libres de errores o que los cambios se compilen y se ejecuten como se espera. Debe verificar los resultados generados por scu y asegurarse de que los cambios cumplen sus expectativas. Es posible que también tenga que modificar más el código que scu ha intentado corregir.

## Opciones del programa de utilidad de conversión de origen (scu)

Hay varias opciones disponibles en el programa de utilidad de conversión de origen (scu) para la conversión de programa fuente.

El mandato scu tiene la sintaxis siguiente:




Descripciones de opciones:

**-7**

Si una línea empieza con un carácter especial, la opción -7 detecta y mueve el carácter a la columna 7 (el área de indicador de formato fijo o ampliado). El carácter especial aquí puede ser un asterisco '*', una barra inclinada '/', un signo de dólar '$', un carácter 'D' seguido por un espacio y' d' seguido por un espacio. Los caracteres que siguen al carácter especial movido se manejan en función de sus posiciones:

- Si los caracteres siguientes están en la columna 2-7, toda la línea se mueve a la derecha hasta que el carácter especial está en la columna 7.
- Si los caracteres siguientes están en el área A o B, permanecen donde están, a menos que haya caracteres en la columna 73 o más allá. Cuando hay caracteres en la columna 73 o más allá, los caracteres se pueden mover a la izquierda al principio del área B.

**Notas:**

- Para un signo de dólar '$', scu emite un error que indica que se necesita una intervención manual para esta línea.
- El carácter especial '-' como primer carácter de una línea se mueve a la columna 7 sólo cuando se especifican las opciones -f y -7 o cuando se especifica la opción -f y '-' está en la columna 1.
- Para los caracteres especiales que no están en la columna 7 o no se mueven a la columna 7, scu maneja estos caracteres como regulares (no especiales).

**-b**
Elimina los espacios en blanco finales.

**-e**
Indica si un archivo de entrada está en formato de origen ampliado de una línea de 252 caracteres. Esta opción permite a `scu` distinguir el formato ampliado del formato fijo predeterminado y convertir el código fuente correctamente.

**-E**
Indica a `scu` que la salida (el origen convertido) no está limitada a las 72 columnas predeterminadas (el formato fijo). `scu` puede ampliar las líneas hasta la longitud máxima de 252 columnas si es necesario.

**-f**
Identifica el origen de entrada como en formato libre. De forma predeterminada, el formato de origen fijo (compatible) permite el código fuente COBOL ejecutable en el Área A (columna 8-11) y el Área B (columna 12-72), con indicadores en la columna 7. Opcionalmente, si especifica `-e`, el área B se amplía a la columna 252 para el archivo de entrada. La opción `-f` hace que `scu` mueva el código fuente COBOL de la columna 1-6 a la columna de indicador, Área A o Área B en función del contenido del código fuente que se va a mover.

La opción `-f` maneja líneas de origen de formato libre que empiezan con caracteres especiales de una de las formas siguientes:

- Si el carácter especial está en la columna 1 (el área de indicador de formato libre), el carácter se mueve a la columna 7 (el área de indicador de formato fijo).
- Si el carácter especial está en la columna 2-11, se mueve a la columna 12 (la columna inicial del Área B).

**Notas:**

- Los caracteres especiales en un archivo de origen de formato libre pueden ser '*', '/', '$', '-', 'D' seguido por el espacio y ' d' seguido por el espacio.
- De forma predeterminada, la opción `-f` no se especifica y un carácter especial al principio de una línea de formato libre se mueve a la columna 12 (columna inicial del Área B) sólo cuando está en la columna 8-11 (Área A). Cuando el carácter especial está en otras columnas, permanece donde está. Un carácter que no es un carácter especial en la columna 7 se deja en blanco.
- Cuando se utiliza la opción `-f` en combinación con la opción `-7` , cualquier carácter especial de la columna 1-6 se mueve a la columna 7. Este uso es similar a la opción `-7` solo, pero incluye '-' como carácter especial.

**-G*< nombre de directorio de salida de libro de copias >***
Arregla archivos COPY y los coloca en el directorio especificado. Para un archivo COPY calificado con un nombre de directorio, el nombre de directorio se conserva como subdirectorio del directorio de archivos COPY especificado. Si no se especifica la opción `-G` , sólo se arregla el archivo fuente principal.

**Nota:** No inserte espacios entre `-G` y *< nombre de directorio de salida de libro de copias >*.

**-h**
Proporciona ayuda básica de `scu` con información sobre las funciones disponibles de `scu`. También puede especificar `-\?` para visualizar la misma información de ayuda que `-h` . Para obtener ayuda más detallada, consulte la página man de `scu` ejecutando el mandato `man scu`.

**-I*< nombre de directorio de entrada de libro de copias >***
Añade la vía de acceso especificada a los directorios en los que se buscarán los libros de copias si no se especifica un nombre de biblioteca o SYSLIB.

**Notas:**

- Esta opción es la letra `I` mayúscula, no la letra l minúscula.
- Solo se permite una única vía de acceso para cada opción `-I` . Para añadir varias vías de acceso, utilice varias opciones de `-I` .

- No inserte espacios entre `-I` y *< nombre de directorio de entrada de libro de copias >*.
- Los archivos COPY que se recuperan de un directorio `-I` , arreglados por `scu` y almacenados en el directorio `-G` se pueden seleccionar especificando `-I` y el mismo directorio `-G` . De este modo, `scu` utiliza una versión fija del archivo de copia en las ejecuciones posteriores en los mismos o distintos archivos de origen principales.

**-L**

Sangra los números de nivel distintos de 01 y 77 al Área B cuando los números de nivel están en el Área A.

**-M**

Emite un código de arreglo de `scu` (por ejemplo, SCU0001) al final de cada línea fija y proporciona una breve descripción y un resumen en la parte inferior del archivo de salida. Cuando se especifica COBOL compatible estándar y se especifica la opción `-M` , se añade un código de arreglo `scu` a las columnas no compilables no utilizadas (a partir de la columna 82) para indicar que se ha cambiado la línea. También se proporciona un resumen para visualizar información de arreglos que está asociada con cada código de arreglo. Los códigos de arreglo y el resumen se proporcionan para la sintaxis y los cambios semánticos, no para los cambios de transformación iniciales. Por ejemplo, cuando especifica `-f` para convertir un archivo de formato libre a una columna 80 fija o formato ampliado, los cambios de línea se realizan pero no recibe un código de arreglo `scu` .

Utilice la numeración de código de arreglo para identificar el nivel de atención necesaria para el mensaje:

*Tabla 3. Niveles de gravedad del mensaje Scu*

| **Rango de códigos de arreglo** | **Gravedad** | **Descripción** |
|---|---|---|
| SCU0001 - SCU1999 | Informativo | Scu espera que no se necesiten más cambios para el arreglo. |
| SCU2000 - SCU3999 | Aviso | Scu espera que sean necesarios más cambios. Por ejemplo, el arreglo de cambiar OCCURS del nivel 01 al nivel 02 puede requerir más cambios relacionados con el arreglo. |
| SCU8000 - SCU8999 | Error | Scu espera que se necesiten más cambios para completar el arreglo. |
| SCX0001 - SCX8999 | Error no corregido | El problema está identificado, pero `scu` no puede solucionar el problema. El código de error SCX*nnnn* corresponde al código de arreglo SCU*nnnn* coincidente. |
| SCX9000 - SCX9999 | Error ignorado | El problema está identificado, pero `scu` no intenta arreglarlo. |

**Nota:** Es posible que Scu no identifique o arregle todas las incompatibilidades que existen en el código.

La lista siguiente proporciona ejemplos de códigos de arreglo de `scu` y los errores corregidos correspondientes:

```
SCU0001 fix for IGYDS0001-W: Add missing space(s).
SCU0004 fix for IGYPS0019-W: Extra and misplaced periods in COBOL source. Scu removes the
extra periods.
SCU1002 fix for IGYGR1080-S: Non-floating point literal is assigned to floating point data
item. Scu converts it to floating point constant notation.
SCU1005 fix for IGYPS2024-S: SET used in place of MOVE. Scu converts SET stmt to MOVE stmt.
SCU1006 fix for IGYPS2094-S: "<>" converted to "NOT = ".
SCU1008 fix for IGYDS0017-E: "01" not in Area A. Scu moves 01 to Area A.
SCU3001 fix for IGYDS1063-E: OCCURS clause in level 01. Scu changes it to level 02 and adds
a dummy 01.
```

```
SCU8001 fix for IGYDS0093-S: RECORD SEQUENTIAL not supported. Scu replaces RECORD
SEQUENTIAL with LINE SEQUENTIAL.
SCX9001 specified for an 02 level data item following an 01 level OCCURS that has been
changed
to an 02 level OCCURS with fix code SCU3001
```

**-N**
Permite a `scu` realizar sólo los cambios de transformación iniciales sin cambios de sintaxis y semánticos, y fuerza que la salida se grabe en la salida estándar.

**-o *< nombre de archivo de salida >***
Especifica el nombre de archivo de salida para el archivo de origen. El archivo de salida se puede calificar con un directorio existente. Por ejemplo, el mandato `scu -o/dirname1/ abc.modified.cbl abc.cbl` guarda el archivo de salida `abc.modified.cbl` en el directorio `/ dirname1` . De forma predeterminada, el archivo de salida se guarda en el directorio actual. Si no se especifica la opción `-o` , el archivo de salida para el archivo de origen sería `abc.scu.cbl`.

**-s**
Elimina los números de secuencia iniciales y finales al dejar en blanco las columnas 1-6 y truncar la línea fuente en la columna 73.

**-t *< ancho_tabla>***
Pasa a `scu` el ancho de pestaña que se utiliza en el código fuente para asegurarse de que los datos convertidos están en columnas correctas. Los caracteres de tabulación que se encuentran antes de una posición de tabulación se sustituyen por espacios suficientes para mover el carácter subsiguiente a la posición de tabulación. El ancho de pestaña predeterminado es 8.

**-t *< tabstop>, ...***
Pasa a `scu` los tabuladores para la conversión. Especifique dos o más tabuladores separados por comas. Los caracteres de tabulación encontrados después de la última posición de tabulación se sustituyen por un único carácter de espacio.

**-v**
Habilita la salida detallada para que la información de errores y arreglos se envíe a STDERR durante la transformación de origen, la comprobación de sintaxis y semántica y el arreglo.

**-V**
Muestra la información de versión de `scu`.

Libros de copias y `scu`:

Es una buena práctica que todos los libros de copias pasen por cambios de transformación antes de que `scu` intente arreglar errores de sintaxis y semánticos, porque actualmente `scu` no realiza automáticamente cambios de transformación en los libros de copias. Primero puede ejecutar `scu` para los libros de copias especificando la opción `-N` con cualquier otra opción de transformación, como por ejemplo `-7`, `-b`, `-e`, `-E`, `-f`, `-L`, `-sy` `-t`. A continuación, ejecute `scu` para los archivos de origen principales y especifique `-I` con el directorio de libro de copias que contiene los libros de copias transformados para el proceso de errores semánticos y de sintaxis.

# Capítulo 4. Migración de Enterprise COBOL for z/OS a COBOL for Linux en x86

Esta información describe algunas áreas que es posible que tenga que tener en cuenta si migra programas de Enterprise COBOL for z/OS a COBOL for Linux en x86.

## Opciones de compilador

COBOL for Linux en x86 no da soporte a las siguientes opciones de compilador Enterprise COBOL.

ADATA, ADV, AFP, ARCH, AWO, BLOCK0, BUFSIZE, CODEPAGE, COPYLOC, COPYRIGHT, DATA, DBCS, DECK, DISPSIGN, DLL, DUMP, EXPORTALL, FASTSRT, HGPR, INITCHECK, INITIAL, INLINE, INTDATE, LANGUAGE, LP, MAXPCF, NAME, NUMCHECK, NUMPROC, OBJECT, OFFSET, OPTFILE, OUTDD, PARMCHECK, CALIFICAR, RENT, RMODE, RULES, SERVICE, SQLCCSID, SEG.

Aunque muchas de las opciones anteriores no tienen ningún equivalente en COBOL for Linux en x86 , las opciones siguientes tienen equivalentes:

- Utilizar ADDR en lugar de LP
- Utilice las opciones de distintivo -dll, -dso o -shared en lugar de DLL
- Utilice -c en combinación con opciones de distintivo -o en lugar de OBJECT

**Tareas relacionadas**
"Getting IBM Enterprise COBOL for z/OS applications to compile" en la publicación *Programming Guide*

**Referencias relacionadas**
"cob2 options" en la publicación *Programming Guide*

## Representación de datos

La representación de datos puede diferir entre Enterprise COBOL y COBOL for Linux en x86.

### Datos binarios

Los compiladores IBM COBOL utilizan la representación nativa de la plataforma al manejar elementos de datos binarios (BINARY, COMP, COMP-4 y COMP-5).

En Linux en x86, los elementos de datos binarios se almacenarán y manipularán en formato little endian, es decir, el dígito menos significativo se encuentra en la dirección más baja. En z/OS, los elementos de datos binarios se almacenarán y manipularán en formato big-endian, es decir, el dígito más significativo está en la dirección más baja.

Al migrar aplicaciones COBOL desde Enterprise COBOL for z/OS a COBOL for Linux en x86, es posible que obtenga resultados inesperados si el programa accede a datos almacenados en formato big-endian como compilador y el tiempo de ejecución tratará los datos como formato little-endian de forma predeterminada.

Puede utilizar la opción `BINARY(BE)` para informar al compilador COBOL for Linux en x86 para manejar elementos de datos BINARY, COMP y COMP-4 en formato big-endian coherentes con Enterprise COBOL for z/OS. Sin embargo, esto tendrá cierta sobrecarga de rendimiento, ya que el compilador necesita convertir los datos a y desde su formato nativo, little-endian. Los elementos de datos COMP-5 no se ven afectados por la opción `BINARY(BE)` o `BINARY(LE)` ya que COMP-5 indica un elemento de datos binario nativo que utiliza la representación nativa de la plataforma. Si utiliza una combinación de COMP-5 y otros tipos de datos BINARY/COMP/COMP-4 en el programa, tenga cuidado al utilizar la opción `BINARY(BE)` . Si un elemento de datos determinado necesita permanecer en representación little-endian (LE) cuando se ha especificado `BINARY(BE)` , utilice la cláusula `NATIVE` en la sentencia USAGE .

**Notas:**

- Si está utilizando IBM MQ, se espera que los parámetros de API estén en formato big endian, por lo que deberá utilizar la opción BINARY (BE) y FLOAT (BE) al trabajar con MQ en Linux.
- IBM Db2 y Oracle Pro *COBOL añaden sus propias áreas de datos en el programa COBOL generado para comunicarse con sus bibliotecas de cliente. Estas bibliotecas de cliente esperan datos en formato little endian nativo en Linux, incluso si dan soporte a datos de host big-endian. Al trabajar con Db2 o Pro *COBOL, debe utilizar el formato binario nativo predeterminado, es decir, BINARY (NATIVE) o BINARY (LE).
- Puesto que IBM MQ espera un formato binario diferente que IBM Db2 y Oracle Pro *COBOL, no se recomienda tener llamadas MQ y SQL en la misma unidad de compilación o compilación por lotes.

## Datos decimales con zona

La representación de signo para datos decimales con zona se basa en ASCII o EBCDIC en función del valor de la opción de compilador `CHAR` (`NATIVE` o `EBCDIC`) y de si la cláusula USAGE se especifica con la frase `NATIVE` . COBOL for Linux en x86 procesa la representación de signo de datos decimales con zona de forma coherente con el proceso que se produce en z/OS cuando la opción de compilador `NUMPROC(NOPFD)` está en vigor.

## Datos decimales empaquetados

La representación de signo para números decimales empaquetados sin signo es diferente entre COBOL for Linux en x86 y Enterprise COBOL. COBOL for Linux en x86 siempre utiliza una porción de signo de `x'C'` para los números decimales empaquetados sin signo. Enterprise COBOL utiliza una porción de signo de `x'F'` para números decimales empaquetados sin signo. Si va a compartir archivos de datos que contienen números decimales empaquetados entre Linux y z/OS, se recomienda que utilice números decimales empaquetados con signo en lugar de números decimales empaquetados sin signo.

## Datos internos de coma flotante (COMP-1, COMP-2)

Puede utilizar la opción de compilador `FLOAT(BE)` para indicar que los elementos de datos de coma flotante internos están en la representación de datos IBM Z (hexadecimal) en contraposición al formato nativo (IEEE).

## Datos nacionales

Los compiladores COBOL de IBM utilizan la endianness nativa de la plataforma al manejar datos nacionales.

En Linux en x86, los elementos de datos nacionales se almacenarán y manipularán en formato little endian UTF-16 . En z/OS, los elementos de datos nacionales se almacenarán y manipularán en formato UTF-16 big-endian.

Al migrar aplicaciones COBOL desde Enterprise COBOL for z/OS a COBOL for Linux en x86, es posible que obtenga resultados inesperados si el programa accede a datos almacenados en formato big-endian porque el compilador y el tiempo de ejecución tratarán los datos como formato little-endian de forma predeterminada.

Puede utilizar la opción `UTF16(BE)` para informar al compilador COBOL for Linux en x86 para manejar elementos de datos nacionales en formato big-endian coherentes con Enterprise COBOL for z/OS. Sin embargo, esto tendrá cierta sobrecarga de rendimiento, ya que el compilador necesita convertir los datos a y desde su formato nativo, little-endian. Si un elemento de datos determinado necesita permanecer en representación little-endian (LE) cuando se ha especificado `UTF16(BE)` , utilice la cláusula NATIVE en la sentencia USAGE.

### Datos EBCDIC y ASCII

Puede especificar la secuencia de clasificación EBCDIC para elementos de datos alfanuméricos utilizando los siguientes elementos de idioma:

- `ALPHABET` Cláusula
- `PROGRAM COLLATING SEQUENCE` Cláusula
- Frase `COLLATING SEQUENCE` de la sentencia `SORT` o `MERGE`

Puede especificar la opción de compilador `CHAR(EBCDIC)` para indicar que los elementos de datos `DISPLAY` están en la representación de datos IBM Z (EBCDIC).

### Determinación de página de códigos para la conversión de datos

Para elementos de datos alfabéticos, alfanuméricos, DBCS y nacionales, la página de códigos fuente utilizada para la conversión implícita de caracteres nativos se determina a partir del entorno local en vigor en tiempo de ejecución.

Para literales alfanuméricos, DBCS y nacionales, la página de códigos fuente utilizada para la conversión implícita de caracteres se determina a partir del entorno local en vigor en el momento de la compilación.

### Series de caracteres DBCS

En COBOL for Linux en x86, las series de caracteres DBCS ASCII no están delimitadas con los caracteres de desplazamiento a teclado estándar y a teclado ideográfico, excepto posiblemente con los caracteres de desplazamiento a teclado estándar y a teclado ideográfico, como se describe a continuación.

Utilice la opción de compilador `SOSI` para indicar que los caracteres de control Linux de estación de trabajo shift-out (X'1E') y shift-in (X'1F') delimitan las series de caracteres DBCS en el programa de origen, incluyendo palabras definidas por el usuario, literales DBCS, literales alfanuméricos, literales nacionales y comentarios. Los caracteres de control de desplazamiento a teclado ideográfico y de desplazamiento a teclado ideográfico de host (X'0E'y X'0F', respectivamente) se suelen convertir a caracteres de control de desplazamiento a teclado ideográfico y de desplazamiento a teclado ideográfico de estación de trabajo cuando se descarga el código fuente COBOL for Linux en x86 , en función del método de descarga que utilice.

El uso de los caracteres de control X'00 'a X'1F' dentro de un literal alfanumérico puede producir resultados imprevisibles.

**Tareas relacionadas**
"Establecimiento del entorno local" en la publicación *Programming Guide*
"Arreglo de diferencias causadas por representaciones de datos" en la publicación *Programming Guide*

**Referencias relacionadas**
"BINARY" en *Guía de programación*
"CHAR" en la *Guía de programación*
"FLOAT" en la *Guía de programación*
"SOSI" en la *Guía de programación*
"UTF16" en la publicación *Programming Guide*

## Variables de entorno de compilador y tiempo de ejecución

COBOL for Linux en x86 reconoce varias variables de compilador y de entorno de ejecución que no se utilizan en Enterprise COBOL, tal como se indica a continuación.

- COBCPYEXT
- COBLSTDIR
- COBOPT
- DB2DBDFT

- DBCS_CODEPAGE
- LANG
- LC_ALL
- LC_COLLATE
- LC_CTYPE
- LC_MESSAGES
- LC_TIME
- LD_LIBRARY_PATH
- *nombre-biblioteca*
- NLSPATH
- SYSLIB
- *nombre-texto*
- TMPDIR
- TZ

Información relacionada:

"Establecimiento de variables de entorno" en *Guía de programación*

## Especificación de archivo

Existen algunas diferencias entre la forma en que COBOL for Linux en x86 maneja los archivos y la forma en que Enterprise COBOL maneja los archivos.

Las diferencias entre COBOL for Linux en x86 y Enterprise COBOL en el manejo de archivos se encuentran en las áreas siguientes:

- Archivos de un solo volumen
- Sufijos de archivo de origen
- Grupos de datos de generación (GDG)
- Concatenación de archivos

**Archivos de un solo volumen:** COBOL for Linux en x86 trata todos los archivos como archivos de un solo volumen. Todas las demás especificaciones de archivo se tratan como comentarios. Esta diferencia afecta a los siguientes elementos: `REEL`, `UNIT`, cláusula `MULTIPLE FILE TAPE` y `CLOSE. . .UNIT/REEL`.

**Sufijos de archivo de origen:** En COBOL for Linux en x86, cuando compila utilizando uno de los mandatos `cob2` , los archivos de origen COBOL que tienen el sufijo .cbl o .cob se pasan al compilador. En Enterprise COBOL, cuando compila en el sistema de archivos z/OS UNIX , sólo se pasan al compilador los archivos que tienen el sufijo .cbl.

**Grupos de datos de generación (GDG):** El soporte de GDG es casi idéntico al soporte de GDG en Enterprise COBOL. Sin embargo, existen diferencias en COBOL for Linux en x86:

- Los conjuntos de datos de generación (GDS) o *archivos de generación* a los que se hace referencia en esta información, están soportados para todas las organizaciones de archivos y modalidades de acceso en todos los sistemas de archivos soportados.
- El soporte de GDG no está integrado en los sistemas de archivos. Se proporciona un programa de utilidad autónomo, `gdgmgr`, para crear y suprimir GDG, gestionar y consultar entradas de GDG, realizar el proceso de límite y conciliar el catálogo de GDG con los archivos existentes.
- La resolución de los nombres de archivo de generación se produce cuando se abren los archivos en lugar de durante la inicialización del trabajo.
- El proceso de límite se realiza cuando se añade una nueva generación a un grupo, en lugar de al terminar el trabajo.

- El rango generacional dentro de cualquier época dada es de 1 a 9999, inclusive, en lugar de estar limitado a 1000. Por lo tanto, las generaciones 0001 y 9999 pueden existir en la misma época.
- Un grupo puede contener 1000 generaciones en lugar de 255.
- El mantenimiento de versiones no está soportado. La versión generada automáticamente es siempre `v00`.

**Concatenación de archivos:** En COBOL for Linux en x86, puede concatenar varios archivos separando los identificadores de archivo con un signo de dos puntos (:). Un archivo COBOL concatenado debe tener una organización secuencial o secuencial de línea, se debe acceder a él secuencialmente y sólo se puede abrir para entrada.

**Conceptos relacionados**
"Sistemas de archivos" en *Guía de programación*
"Grupos de datos de generación" en *Guía de programación*

**Tareas relacionadas**
"Concatenación de archivos" en *Guía de programación*
"Compilación desde la línea de mandatos" en la publicación *Programming Guide*

**Referencias relacionadas**
"Limitar el proceso de grupos de datos de generación" en la publicación *Programming Guide*

## Comunicación interlingüística (ILC)

ILC está disponible con programas C/C++ .

Estas son las diferencias en el comportamiento de ILC en Linux en x86 en comparación con el uso de ILC en z/OS con Language Environment:

- Existen diferencias en el comportamiento de terminación cuando se utiliza un COBOL `STOP RUN` o una salida C () .
- No hay un manejo coordinado de condiciones con COBOL for Linux en x86. Evite utilizar un C longjmp () que cruce programas COBOL.
- Con Enterprise COBOL, el primer programa que se invoca dentro del proceso y que está habilitado para Language Environment se considera el programa "principal". Con COBOL for Linux en x86, el primer programa COBOL invocado en el proceso se considera el programa principal por COBOL. Esta diferencia afecta a la semántica del lenguaje que es sensible a la definición de la unidad de ejecución (la unidad de ejecución que empieza con un programa principal). Por ejemplo, un `STOP RUN` da como resultado la devolución del control al invocador del programa principal, que en un entorno de lenguaje mixto puede ser diferente como se ha indicado anteriormente.

**Conceptos relacionados**
"Llamada entre programas COBOL y C/C++" en *Guía de programación*
"Preinicialización del entorno de ejecución COBOL" en la publicación *Programming Guide*

## Entrada y salida

COBOL for Linux en x86 da soporte a la entrada y salida para archivos secuenciales, relativos e indexados con los sistemas de archivos Db2, MONGO, SdU, SFS y STL, sistemas de archivos y también da soporte a la entrada y salida para archivos secuenciales de con el sistema de archivos QSAM y el sistema de archivos RSD.

La entrada y salida secuencial de línea está soportada por el soporte de archivo continuo de bytes nativo del sistema operativo.

Los tamaños y valores de la información de estado de archivo devuelta pueden variar en función del sistema de archivos que se utilice.

COBOL for Linux en x86 no proporciona soporte directo para unidades de cintas o unidades de disquetes.

**Conceptos relacionados**
"Sistemas de archivos" en la publicación *Programming Guide*
"Organización de archivos secuenciales de línea" en la *Guía de programación*

**Tareas relacionadas**
"Utilización de claves de estado de archivo" en la publicación *Programming Guide*
"Utilización de códigos de estado del sistema de archivos" en la publicación *Programming Guide*

## Opciones de tiempo de ejecución

COBOL for Linux en x86 no reconoce las siguientes opciones de tiempo de ejecución de Enterprise COBOL y las trata como no válidas: `AIXBLD`, `ALL31`, `CBLPSHPOP`, `CBLQDA`, `COUNTRY`, `HEAP`, `MSGFILE`, `NATLANG`, `SIMVRD`y `STACK`.

Con Enterprise COBOL, puede utilizar la opción de tiempo de ejecución `STORAGE` para inicializar COBOL `WORKING-STORAGE`. Con COBOL for Linux en x86, utilice la opción de compilador `WSCLEAR` .

**Referencias relacionadas**
"Variables de entorno de ejecución" en *Guía de programación*
"Variables de entorno comunes de compilador y tiempo de ejecución" en la publicación *Programming Guide*
"WSCLEAR" en *Guía de programación*

## Tamaño de línea de código fuente

En COBOL for Linux en x86, las líneas fuente COBOL pueden tener longitudes variables. Una línea fuente finaliza cuando se encuentra un carácter de control de nueva línea o cuando se ha alcanzado la longitud máxima de línea.

En Enterprise COBOL, cada línea de origen tiene la misma longitud.

**Referencias relacionadas**
"SRCFORMAT" en la publicación *Programming Guide*

## Elementos de lenguaje

La tabla siguiente lista elementos de lenguaje que son diferentes entre compiladores de Enterprise COBOL y COBOL for Linux en x86 , y donde sea posible ofrece consejos sobre cómo manejar estas diferencias en programas COBOL for Linux en x86 .

Muchas cláusulas y frases COBOL que son válidas en Enterprise COBOL se comprueban en la sintaxis pero no tienen ningún efecto en la ejecución de programas COBOL for Linux en x86 . Estas cláusulas y frases deben tener un efecto mínimo en las aplicaciones existentes que descargue. COBOL for Linux en x86 reconoce la mayor parte de la sintaxis del lenguaje Enterprise COBOL incluso si dicha sintaxis no tiene ningún efecto funcional.

*Tabla 4. Diferencias de lenguaje entre Enterprise COBOL for z/OS y COBOL for Linux en x86*

| **Elemento de idioma** | **Implementación o restricción de COBOL for Linux en x86** |
|---|---|
| `ACCEPT` sentencia | Si el programa Enterprise COBOL espera ddnames como los destinos de las sentencias `ACCEPT` , defina estos destinos utilizando variables de entorno equivalentes con valores establecidos en los nombres de archivo adecuados. En COBOL for Linux en x86, *nombre-entorno* y el valor de variable de entorno asociado, si se establece, determine la identificación del archivo. |
| `APPLY WRITE-ONLY` Cláusula | Sintaxis comprobada, pero no tiene ningún efecto en la ejecución del programa en COBOL for Linux en x86 |

| *Tabla 4. Diferencias de lenguaje entre Enterprise COBOL for z/OS y COBOL for Linux en x86 (continuación)* | |
|---|---|
| **Elemento de idioma** | **Implementación o restricción de COBOL for Linux en x86** |
| ASSIGN Cláusula | COBOL for Linux en x86 utiliza una sintaxis diferente y una correlación con el nombre de archivo del sistema basada en *nombre-asignación*. ASSIGN. . .USING *nombre-datos* no está soportado en Enterprise COBOL. |
| BLOCK CONTAINS Cláusula | Sintaxis comprobada, pero no tiene ningún efecto en la ejecución del programa en COBOL for Linux en x86 |
| CALL sentencia | Un nombre de archivo como argumento CALL no está soportado en COBOL for Linux en x86. |
| CLOSE sentencia | Las frases siguientes se comprueban con la sintaxis, pero no tienen ningún efecto en la ejecución del programa en COBOL for Linux en x86: FOR REMOVAL, WITH NO REWINDy UNIT/REEL. Evite el uso de estas frases en programas que están pensados para ser portátiles. |
| CODE-SET Cláusula | Sintaxis comprobada, pero no tiene ningún efecto en la ejecución del programa en COBOL for Linux en x86 |
| DATA RECORDS Cláusula | Sintaxis comprobada, pero no tiene ningún efecto en la ejecución del programa en COBOL for Linux en x86 |
| DISPLAY sentencia | Si el programa Enterprise COBOL espera ddnames como los destinos de las sentencias DISPLAY , defina estos destinos utilizando variables de entorno equivalentes con valores establecidos en los nombres de archivo adecuados. En COBOL for Linux en x86, *nombre-entorno* y el valor de variable de entorno asociado, si se establece, determine la identificación del archivo. |
| DYNAMIC LENGTH Cláusula | Los elementos elementales de longitud dinámica no están soportados actualmente en COBOL for Linux en x86. |
| Estado de archivo *data-name-1* | Algunos valores y significados para el estado de archivo 9*x* son diferentes en Enterprise COBOL que en COBOL for Linux en x86. |
| Estado de archivo *data-name-8* | El formato y los valores son diferentes en función de la plataforma y el sistema de archivos. |
| INDEX elementos de datos | En Enterprise COBOL, los elementos de datos INDEX se definen implícitamente como 4 bytes. En programas COBOL for Linux en x86 compilados con ADDR(32), su tamaño es de 4 bytes; con ADDR(64), su tamaño es de 8 bytes. |
| Sentencia INVOKE | COBOL for Linux en x86 no da soporte a la escritura de programas orientados a objetos, a la creación de instancias de objeto de una clase COBOL o Java™ o a la invocación de un método definido en una clase COBOL o Java . |
| Sentencias JSON GENERATE y JSON PARSE | JSON no está soportado actualmente en COBOL for Linux en x86. |
| LABEL RECORDS Cláusula | Las frases LABEL RECORD IS *nombre-datos*, USE. . .AFTER. . .LABEL PROCEDUREy GO TO MORE-LABELS se comprueban con la sintaxis, pero no tienen ningún efecto en la ejecución del programa en COBOL for Linux en x86. Se emite un aviso si utiliza alguna de estas frases. No se llama a los declarativos de etiqueta de usuario en tiempo de ejecución. No puede portar programas que dependan del proceso de etiqueta de usuario al que z/OS QSAM da soporte. |
| MULTIPLE FILE TAPE | La sintaxis se ha comprobado, pero no tiene ningún efecto en la ejecución del programa en COBOL for Linux en x86. En la estación de trabajo Linux , todos los archivos se tratan como archivos de un solo volumen. |
| OBJECT REFERENCE elementos de datos | Los elementos de datos OBJECT REFERENCE no están soportados en COBOL for Linux en x86. |

| *Tabla 4. Diferencias de lenguaje entre Enterprise COBOL for z/OS y COBOL for Linux en x86 (continuación)* | |
|---|---|
| **Elemento de idioma** | **Implementación o restricción de COBOL for Linux en x86** |
| OPEN sentencia | Las frases siguientes se comprueban con la sintaxis, pero no tienen ningún efecto en la ejecución del programa en COBOL for Linux en x86: REVERSED y WITH NO REWIND. |
| PASSWORD Cláusula | Sintaxis comprobada, pero no tiene ningún efecto en la ejecución del programa en COBOL for Linux en x86 |
| Elementos de datos POINTER, PROCEDURE-POINTERy FUNCTION-POINTER | En Enterprise COBOL, los elementos de datos POINTER y FUNCTION-POINTER se definen implícitamente como 4 bytes ya que es el registro especial ADDRESS OF; PROCEDURE-POINTER los elementos de datos se definen implícitamente como 8 bytes. En programas COBOL for Linux en x86 compilados con ADDR(32), el tamaño de cada uno de estos elementos es de 4 bytes; con ADDR(64), su tamaño es de 8 bytes. |
| READ. . .PREVIOUS | Solo en COBOL for Linux en x86 , le permite leer el registro anterior para archivos relativos o indexados con la modalidad de acceso DYNAMIC |
| RECORD CONTAINS Cláusula | La cláusula RECORD CONTAINS *n* CHARACTERS se acepta con una excepción: RECORD CONTAINS 0 CHARACTERS se comprueba la sintaxis, pero no tiene ningún efecto en la ejecución del programa en COBOL for Linux en x86. |
| RECORDING MODE Cláusula | La sintaxis se ha comprobado, pero no tiene ningún efecto en la ejecución del programa en COBOL for Linux en x86 para los archivos relativos, indexados y secuenciales de línea. RECORDING MODE U se comprueba la sintaxis, pero no tiene ningún efecto en la ejecución del programa para archivos secuenciales. |
| RERUN Cláusula | Sintaxis comprobada, pero no tiene ningún efecto en la ejecución del programa en COBOL for Linux en x86 |
| RESERVE Cláusula | Sintaxis comprobada, pero no tiene ningún efecto en la ejecución del programa en COBOL for Linux en x86 |
| SAME AREA Cláusula | Sintaxis comprobada, pero no tiene ningún efecto en la ejecución del programa en COBOL for Linux en x86 |
| SAME SORT Cláusula | Sintaxis comprobada, pero no tiene ningún efecto en la ejecución del programa en COBOL for Linux en x86 |
| SHIFT-IN, SHIFT-OUT registros especiales | El compilador COBOL for Linux en x86 emite un mensaje de nivel E si encuentra estos registros a menos que la opción de compilador CHAR(EBCDIC) esté en vigor. |
| Registro especial de SORT-CONTROL | La definición implícita y el contenido de este registro especial difieren entre el host y la estación de trabajo COBOL. |
| Registro especial de SORT-CORE-SIZE | El contenido de este registro especial difiere entre COBOL de host y estación de trabajo. |
| Registro especial de SORT-FILE-SIZE | La sintaxis se ha comprobado, pero no tiene ningún efecto en la ejecución del programa en COBOL for Linux en x86. Los valores de este registro especial no se utilizan. |
| Registro especial de SORT-MESSAGE | Sintaxis comprobada, pero no tiene ningún efecto en la ejecución del programa en COBOL for Linux en x86 |
| Registro especial de SORT-MODE-SIZE | La sintaxis se ha comprobado, pero no tiene ningún efecto en la ejecución del programa en COBOL for Linux en x86. Los valores de este registro especial no se utilizan. |

| *Tabla 4. Diferencias de lenguaje entre Enterprise COBOL for z/OS y COBOL for Linux en x86 (continuación)* | |
|---|---|
| **Elemento de idioma** | **Implementación o restricción de COBOL for Linux en x86** |
| SORT MERGE AREA Cláusula | Sintaxis comprobada, pero no tiene ningún efecto en la ejecución del programa en COBOL for Linux en x86 |
| START... | En COBOL for Linux en x86, se permiten los siguientes operadores relacionales: IS LESS THAN, IS <, IS NOT GREATER THAN, IS NOT >, IS LESS THAN OR EQUAL TO, IS <=. |
| STOP RUN | No soportado en programas multihebra de COBOL for Linux en x86 ; se puede sustituir en un programa multihebra con una llamada a la función C exit () |
| Frase UTF-8 de la cláusula USAGE y el símbolo 'U' PICTURE | La clase de datos UTF-8 y la categoría de datos UTF-8 no están soportadas actualmente en COBOL for Linux en x86. |
| WRITE sentencia | En COBOL for Linux en x86, si especifica WRITE. . .ADVANCING con los nombres de entorno C01 a C12 o S01 a S05, una línea es avanzada. |
| XML PARSE sentencia | En los programas Enterprise COBOL compilados utilizando la opción de sólo host XMLPARSE(XMLSS), hay disponibles sintaxis adicional (la frase ENCODING y la frase RETURNING NATIONAL ) y registros especiales para el proceso de espacio de nombres que no están disponibles con COBOL for Linux en x86. |
| Nombres conocidos para el entorno de plataforma | Los nombres siguientes se identifican de forma diferente: nombre de programa, nombre de texto, nombre de biblioteca, nombre de asignación, nombre de archivo en el registro especial de SORT-CONTROL , nombre base, identificación de destino de DISPLAY o ACCEPT y nombres dependientes del sistema. |

# Productos complementarios

COBOL Report Writer no está disponible en Linux en x86.

El precompilador COBOL Report Writer se utiliza en z/OS para compilar aplicaciones que contienen sentencias Report Writer, o para convertir permanentemente sentencias Report Writer en sentencias Enterprise COBOL válidas. Encontrará más información sobre COBOL Report Writer en la documentación de Enterprise COBOL for z/OS Documentation.

Si está migrando programas COBOL desde z/OS a Linux en x86 y estos programas utilizan el precompilador COBOL Report Writer , en primer lugar utilice el precompilador para convertir permanentemente sentencias Report Writer a sentencias Enterprise COBOL y, a continuación, migre los programas COBOL postprocesados a Linux en x86. Es posible que sea necesario volver a procesar los cambios futuros en los programas que requieren actualizaciones en las secciones relacionadas con informes en z/OS y, a continuación, moverlos a Linux en x86.

# Obtención de aplicaciones IBM Enterprise COBOL for z/OS para compilar

Si mueve programas Enterprise COBOL de un sistema IBM Z a un sistema Linux en x86 y los compila utilizando COBOL for Linux en x86, debe elegir las opciones de compilador correctas y tener en cuenta las características de lenguaje que difieren de IBM Enterprise COBOL for z/OS. También puede utilizar la sentencia COPY para ayudar a los programas de puerto.

## Acerca de esta tarea

**Elección de las opciones de compilador correctas:** Para obtener información adicional sobre las opciones de compilador Enterprise COBOL que afectan a la portabilidad, consulte la referencia relacionada sobre las opciones de compilador.

**Permitir características de lenguaje de Enterprise COBOL:** varias características de lenguaje que son válidas en programas Enterprise COBOL pueden crear errores o resultados imprevisibles cuando se compilan con COBOL for Linux en x86. Para obtener detalles, consulte la referencia relacionada sobre los elementos de lenguaje.

**Utilización de la sentencia COPY para ayudar a los programas de puerto:** en muchos casos, puede evitar posibles problemas de portabilidad utilizando la sentencia COPY para aislar el código específico de la plataforma. Por ejemplo, puede incluir código específico de la plataforma en una compilación para una plataforma determinada y excluirla de la compilación para una plataforma diferente. También puede utilizar la frase `COPY  REPLACING` para cambiar globalmente elementos de código fuente no portátiles, como nombres de archivo.

**Tareas relacionadas**
"Establecimiento de variables de entorno" en la publicación *Programming Guide*

**Referencias relacionadas**
Opciones de compilador
“Elementos de lenguaje” en la página 34
"sentencia COPY" en la publicación *Consulta de lenguaje*

# Obtención de aplicaciones IBM Enterprise COBOL for z/OS para ejecutar: visión general

Después de descargar un programa Enterprise COBOL y compilarlo correctamente utilizando COBOL for Linux en x86, el paso siguiente es ejecutar el programa. En muchos casos, puede obtener los mismos resultados que en IBM z/OS sin modificar mucho el origen.

## Acerca de esta tarea

Para evaluar si se debe modificar el origen, debe saber cómo arreglar los elementos y el comportamiento del lenguaje COBOL que varían debido a la arquitectura de hardware o software subyacente.

**Tareas relacionadas**
“Arreglo de diferencias causadas por representaciones de datos” en la página 38
“Arreglo de diferencias de entorno que afectan a la portabilidad” en la página 41
“Arreglo de diferencias causadas por elementos de lenguaje” en la página 41

# Arreglo de diferencias causadas por representaciones de datos

Para garantizar el mismo comportamiento para los programas, debe comprender las diferencias en determinadas formas de representar los datos y realizar la acción adecuada.

## Acerca de esta tarea

Los datos de caracteres se pueden representar de forma diferente, en función de la cláusula USAGE que describe los elementos de datos y el entorno local que está en vigor en tiempo de ejecución. COBOL almacena decimales empaquetados firmados de la misma forma en ambos Linux en x86 y IBM z/OS. Sin embargo, los datos binarios, decimales externos, de coma flotante y decimales empaquetados sin signo se representan de forma predeterminada de forma diferente.

La mayoría de los programas se comportan igual en IBM z/OS y Linux en x86 independientemente de la representación de datos.

**Tareas relacionadas**
“Manejo de diferencias en caracteres SBCS ASCII y SBCS EBCDIC” en la página 39
“Manejo de diferencias en datos IEEE y hexadecimales” en la página 39
“Manejo de diferencias en series ASCII multibyte y EBCDIC DBCS” en la página 40

**Referencias relacionadas**
“Representación de datos” en la página 29

## Manejo de diferencias en caracteres SBCS ASCII y SBCS EBCDIC

Para evitar problemas con la representación de datos diferente entre caracteres ASCII y EBCDIC, utilice la opción de compilador `CHAR(EBCDIC)` .

### Acerca de esta tarea

COBOL for Linux en x86 utiliza el juego de caracteres ASCII y Enterprise COBOL for z/OS utiliza el juego de caracteres EBCDIC. Por lo tanto, la mayoría de los caracteres tienen un valor hexadecimal diferente, tal como se muestra en la tabla siguiente.

*Tabla 5.* ***Caracteres ASCII contrastados con EBCDIC***

| **Carácter** | **Valor hexadecimal si ASCII** | **Valor hexadecimal si EBCDIC** |
|---|---|---|
| De '0' a '9' | X'30 'a X'39' | X'F0'a X'F9' |
| "a" | X'61 " | X'81 " |
| "A" | X'41 " | X'C1' |
| vacío | X'20 " | X'40 " |

Además, el código que depende de los valores hexadecimales EBCDIC de los datos de tipo carácter probablemente falla cuando los datos de tipo carácter tienen valores ASCII, como se muestra en la tabla siguiente.

*Tabla 6.* ***Comparaciones ASCII contrastadas con EBCDIC***

| **Comparación** | **Evaluación si ASCII** | **Evaluación si EBCDIC** |
|---|---|---|
| 'a' < 'A' | Falso | Verdadero |
| 'A' < '1' | Falso | Verdadero |
| *x* > = '0' | Si es true, no indica si *x* es un dígito | Si es true, *x* es probablemente un dígito |
| *x* = X'40 ' | No prueba si *x* es un espacio en blanco | Prueba si *x* es un espacio en blanco |

Debido a estas diferencias, los resultados de la ordenación de series de caracteres son diferentes entre EBCDIC y ASCII. Para muchos programas, estas diferencias no tienen ningún efecto, pero debe tener en cuenta los posibles errores lógicos si el programa depende de la secuencia exacta en la que se ordenan algunas series de caracteres. Si el programa depende de la secuencia de clasificación EBCDIC y lo está portando a la estación de trabajo, puede obtener la secuencia de clasificación EBCDIC utilizando `PROGRAM COLLATING SEQUENCE IS EBCDIC` o la opción de compilador `COLLSEQ(EBCDIC)` .

**Referencias relacionadas**
"CHAR" en la publicación *Programming Guide*
"COLLSEQ" en la *Guía de programación*

## Manejo de diferencias en datos IEEE y hexadecimales

Para evitar la mayoría de problemas con la representación diferente entre los datos de coma flotante IEEE y hexadecimal, utilice la opción de compilador `FLOAT(BE)` .

## Acerca de esta tarea

COBOL for Linux on x86 representa datos de coma flotante utilizando el formato IEEE. Enterprise COBOL for z/OS utiliza el formato hexadecimal IBM Z . La tabla siguiente resume las diferencias entre IEEE de coma flotante normalizado y hexadecimal normalizado para datos USAGE COMP-1 y datos USAGE COMP-2 .

*Tabla 7.* ***IEEE contrastado con hexadecimal***

| **Especificación** | **IEEE para datos de COMP-1** | **Hexadecimal para datos COMP-1** | **IEEE para datos de COMP-2** | **Hexadecimal para datos COMP-2** |
|---|---|---|---|---|
| Rango | 1.17E-38* a 3.37E+38* | 5.4E-79* a 7.2E+75* | 2.23E-308* a 1.67E+308* | 5.4E-79* a 7.2E+75* |
| Representación de exponente | 8 bits | 7 bits | 11 bits | 7 bits |
| Representación de Mantissa | 23 bits | 24 bits | 53 bits | 56 bits |
| Dígitos de precisión | 6 dígitos | 6 dígitos | 15 dígitos | 16 dígitos |
| * Indica que el valor puede ser positivo o negativo. | | | | |

Para la mayoría de los programas, estas diferencias no deberían crear problemas. Sin embargo, tenga cuidado al portar si el programa depende de la representación hexadecimal de los datos.

**Consideración sobre el rendimiento:** En general, la representación de coma flotante de IBM Z hace que un programa se ejecute más lentamente porque el software debe simular la semántica de las instrucciones de hardware de IBM Z . Esta es una consideración especialmente si la opción de compilador FLOAT(BE) está en vigor y un programa tiene un gran número de cálculos de coma flotante.

"Ejemplos: datos numéricos y representación interna" en la publicación *Programming Guide*

**Referencias relacionadas**
"FLOAT" en la *Guía de programación*

# Manejo de diferencias en series ASCII multibyte y EBCDIC DBCS

Para obtener el comportamiento de Enterprise COBOL para elementos de datos alfanuméricos que contienen caracteres DBCS, utilice las opciones de compilador CHAR(EBCDIC) y SOSI . Para evitar problemas con la representación de datos diferente entre caracteres DBCS ASCII y DBCS EBCDIC, utilice la opción de compilador CHAR(EBCDIC) .

## Acerca de esta tarea

En elementos de datos alfanuméricos, las series de caracteres de doble byte de Enterprise COBOL (que contienen caracteres DBCS EBCDIC) se incluyen entre códigos de desplazamiento, y las series de caracteres de varios bytes de COBOL for Linux en x86 (que contienen caracteres DBCS ASCII, UTF-8o EUC ) no se incluyen entre códigos de desplazamiento. Los valores hexadecimales utilizados para representar los mismos caracteres también son diferentes.

En elementos de datos DBCS, las series de caracteres de doble byte de Enterprise COBOL no están entre códigos de desplazamiento, pero los valores hexadecimales utilizados para representar caracteres son diferentes de los valores hexadecimales utilizados para representar los mismos caracteres en series de varios bytes de COBOL for Linux en x86 .

Para la mayoría de los programas, estas diferencias no deberían dificultar la adaptación. Sin embargo, si el programa depende del valor hexadecimal de una serie de varios bytes, o espera que una serie de caracteres alfanuméricos contenga una mezcla de caracteres de un solo byte y caracteres de varios bytes, tenga cuidado en sus prácticas de codificación.

**Referencias relacionadas**
"CHAR" en la publicación *Programming Guide*
"SOSI" en la *Guía de programación*

# Arreglo de diferencias de entorno que afectan a la portabilidad

Las diferencias en nombres de archivo y códigos de control entre Linux en x86 y las plataformas IBM z/OS pueden afectar a la portabilidad de los programas.

## Acerca de esta tarea

Los convenios de denominación de archivos en Linux en x86 son muy diferentes de los de IBM z/OS. Esta diferencia puede afectar a la portabilidad si utiliza nombres de archivo en los programas fuente COBOL. El siguiente nombre de archivo, por ejemplo, es válido en la estación de trabajo basada en Linux en x86 pero no en IBM z/OS (excepto en el sistema de archivos z/OS UNIX ):

```
/users/joesmith/programs/cobol/myfile.cbl
```

**Distinción entre mayúsculas y minúsculas:** A diferencia de z/OS, Linux distingue entre mayúsculas y minúsculas. Los nombres utilizados en los programas fuente (por ejemplo, nombres de archivo en mayúsculas) deben nombrarse correctamente en los directorios de archivos Linux .

Algunos caracteres que no tienen un significado concreto en z/OS se interpretan como caracteres de control mediante Linux. Esta diferencia puede llevar a un proceso incorrecto de archivos de texto ASCII. Los archivos no deben contener ninguno de los caracteres siguientes:

- X'0A' (LF: salto de línea)
- X'0D' (CR: retorno de carro)
- X'1A' (EOF: fin de archivo)

Si utiliza códigos de control dependientes de dispositivo (específicos de la plataforma) en sus programas o archivos, estos códigos de control pueden causar problemas cuando intenta portar los programas o archivos a plataformas que no soportan los códigos de control. Al igual que con el resto de código específico de la plataforma, es mejor aislar dicho código tanto como sea posible para que pueda sustituirlo fácilmente cuando mueva la aplicación a otra plataforma.

# Arreglo de diferencias causadas por elementos de lenguaje

En general, puede esperar que los programas COBOL portátiles se comporten de la misma forma en Linux que en z/OS. Sin embargo, tenga en cuenta las diferencias en los valores de estado de archivo utilizados en el proceso de E/S.

## Acerca de esta tarea

Si el programa responde a elementos de datos de estado de archivo, preocupe dos problemas, en función de si el programa se graba para responder al primer o segundo elemento de datos de estado de archivo:

- Si el programa responde al primer elemento de datos de estado de archivo (*data-name-1*), tenga en cuenta que los valores devueltos en el rango 9*n* dependen de la plataforma. Si el programa se basa en la interpretación de un valor de 9*n* determinado (por ejemplo, 97), no espere que el valor tenga el mismo significado en Linux que tiene en z/OS. En su lugar, revise el programa para que responda a cualquier valor de 9*n* como una anomalía de E/S genérica.
- Si el programa responde al segundo elemento de datos de estado de archivo (*data-name-8*), tenga en cuenta que los valores devueltos dependen tanto de la plataforma como del sistema de archivos. Por ejemplo, el sistema de archivos STL devuelve valores con una estructura de registro diferente en Linux que el sistema de archivos VSAM en z/OS. Si el programa se basa en la interpretación del segundo elemento de datos de estado de archivo, es probable que el programa no sea portátil.

**Tareas relacionadas**
"Utilización de claves de estado de archivo" en la publicación *Programming Guide*
"Utilización de códigos de estado del sistema de archivos" en la publicación *Programming Guide*

**Referencias relacionadas**
"cláusula FILE STATUS" en la publicación *Programming Guide*
"Clave de estado de archivo" en la publicación *Programming Guide*

# Escritura de código para ejecutarse con IBM Enterprise COBOL for z/OS

Puede utilizar COBOL for Linux en x86 para desarrollar nuevas aplicaciones y aprovechar las ganancias de productividad y la mayor flexibilidad de utilizar la estación de trabajo de Linux en el sistema x86. Sin embargo, al desarrollar programas COBOL, debe evitar utilizar características que no estén soportadas por IBM Enterprise COBOL for z/OS.

## Acerca de esta tarea

**Características de lenguaje:** COBOL for Linux en x86 da soporte a varias características de lenguaje que no están soportadas por Enterprise COBOL. A medida que escribe código en Linux en x86 que está pensado para ejecutarse en z/OS, evite utilizar estas características:

- Nombres de página de códigos como argumentos para las funciones intrínsecas `DISPLAY-OF` y `NATIONAL-OF`
- Sentencia `READ` utilizando la frase `PREVIOUS`
- Sentencia `START` utilizando <, <=o `NOT` > en la frase KEY
- `>>CALLINTERFACE` Directiva de compilador

**Opciones de compilador:** varias opciones de compilador no están disponibles en Enterprise COBOL. No utilice ninguna de las siguientes opciones de compilador en el código fuente si tiene previsto portar el código a z/OS:

- `BINARY(NATIVE)`
- `CALLINT` (tratado como un comentario)
- `CHAR(NATIVE)`
- `FLOAT(NATIVE)`

**Nombres de archivo:** Tenga en cuenta la diferencia en los convenios de denominación de archivos entre Linux y los sistemas de archivos de host. Evite codificar los nombres de los archivos en los programas fuente. En su lugar, utilice nombres nemotécnicos que defina en cada plataforma y correlaciónelos a su vez con nombres de definición de datos de sistema principal o variables de entorno. A continuación, puede compilar el programa para acomodar los cambios en los nombres de archivo sin tener que cambiar el código fuente.

Específicamente, considere cómo hacer referencia a los archivos en los siguientes elementos de lenguaje:

- Nombres de destino `ACCEPT` o `DISPLAY`
- `ASSIGN` Cláusula
- Sentencia `COPY` (*text-name* o *library-name*)

**Sufijos de archivo:** En COBOL for Linux en x86, cuando compila utilizando uno de los mandatos `cob2` , los archivos de origen COBOL que tienen el sufijo .cbl o .cob se pasan al compilador. En COBOL de sistema principal, cuando compila en el sistema de archivos z/OS UNIX , sin embargo, sólo se pasan al compilador los archivos que tienen el sufijo .cbl.

**Programas anidados:** Los programas multihebra en el sistema principal deben ser recursivos. Por lo tanto, evite codificar programas anidados si tiene previsto portar los programas al sistema principal y habilitarlos para que se ejecuten en un entorno multihebra.

# Capítulo 5. Migración de COBOL for AIX a COBOL for Linux en x86

Esta información describe algunas áreas que es posible que tenga que tener en cuenta si migra programas de COBOL para AIX a COBOL for Linux en x86.

## Opciones de compilador

COBOL for Linux en x86 no da soporte a las siguientes opciones de compilador COBOL for AIX .

**ADATA**
La opción ADATA no está soportada actualmente. Utilice el **NOADATA** predeterminado por ahora.

**ARCH**
Esta opción la utiliza COBOL for AIX para dirigirse a distintas arquitecturas de máquina Power . No está soportado en Linux porque COBOL for Linux en x86 utiliza el conjunto de instrucciones x86 más eficiente disponible actualmente.

**ENTRYINT**
Esta opción se trata como un comentario en AIXy no está soportada en Linux.

**DUMP**
El uso de esta opción generará un mensaje de aviso pero no terminará la compilación.

**LIB**
El uso de esta opción generará un mensaje de aviso pero no terminará la compilación.

**MAXMEM**
El uso de esta opción generará un mensaje de aviso pero no terminará la compilación.

**SIZE**
El uso de esta opción generará un mensaje de aviso pero no terminará la compilación.

COBOL for Linux en x86 no da soporte a las siguientes opciones de distintivo COBOL for AIX :

**-cmain**

En Linux, esta opción se acepta y se ignora.

Con COBOL for AIX, la opción **-cmain** sólo tiene efecto si también especifica **-host**. Cuando se utiliza **-host** , el compilador necesita un programa pseudo principal para convertir los argumentos de EBCDIC a ASCII y, a continuación, llamar al programa COBOL. Si tiene un archivo de objeto C o PL/I que contiene una rutina principal, la opción **-cmain** informa a cob2 de que no se enlace con el pseudo principal y convierte al archivo de objeto C o PL/I que contiene una rutina principal en el punto de entrada principal del archivo ejecutable.

Con COBOL for Linux en x86, cob2 siempre se enlaza con el pseudo principal y fuerza que sea el punto de entrada del ejecutable. Todavía puede tener su propio programa C llamado `main`, y se llamará desde el pseudo-principal. Si proporciona su propio main (), debe comprender que se le está pasando una única lista de parámetros de estilo z/OS con la serie terminada en nulo.

```
struct plist {
    uint16_t len;
    uint8_t  str[1025];
};
```

**-p y -pg**
Estas opciones son opciones de creación de perfiles para utilizarlas con tprof en AIX. En Linux, utilice Valgrind en su lugar. No se necesitan instrumentación ni cambios adicionales en el programa COBOL para utilizar Valgrind.

**Referencias relacionadas**
"cob2 options" en la publicación *Programming Guide*

# Representación de datos

La representación de datos puede diferir entre COBOL for AIX y COBOL for Linux en x86.

## Datos binarios

Los compiladores IBM COBOL utilizan la representación nativa de la plataforma al manejar elementos de datos binarios (BINARY, COMP, COMP-4 y COMP-5).

En Linux x86, los elementos de datos binarios se almacenarán y manipularán en formato little-endian (dígito menos significativo en la dirección más baja). En AIX, los elementos de datos binarios se almacenarán y manipularán en formato big-endian (dígito más significativo en la dirección más baja).

Al migrar aplicaciones COBOL de COBOL para AIX a COBOL for Linux en x86, es posible que obtenga resultados inesperados si el programa accede a datos almacenados en formato big-endian como compilador y el tiempo de ejecución tratará los datos como formato little-endian de forma predeterminada..

Puede utilizar la opción `BINARY(BE)` para informar al compilador COBOL for Linux en x86 para manejar elementos de datos BINARY, COMP y COMP-4 en formato big-endian coherentes con COBOL for AIX, sin embargo, esto tendrá cierta sobrecarga de rendimiento ya que el compilador necesita convertir los datos a y desde su formato nativo, little-endian. Los elementos de datos COMP-5 no se ven afectados por la opción BINARY (BE) o BINARY (LE) ya que COMP-5 indica un elemento de datos binario nativo que utiliza la representación nativa de la plataforma. Si utiliza una combinación de COMP-5 y otros tipos de datos BINARY/COMP/COMP-4 en el programa, tenga cuidado al utilizar la opción BINARY (BE). Si un elemento de datos determinado necesita permanecer en la representación little-endian (LE) cuando se ha especificado BINARY (BE), utilice la cláusula NATIVE en la sentencia USAGE.

**Nota:**

- Si está utilizando IBM MQ, se espera que los parámetros de API estén en formato big endian, por lo que deberá utilizar la opción BINARY (BE) y FLOAT (BE) al trabajar con MQ en Linux.
- IBM Db2y Oracle Pro *COBOL añaden sus propias áreas de datos en el programa COBOL generado para comunicarse con sus bibliotecas de cliente. Estas bibliotecas de cliente esperan datos en formato little endian nativo en Linux, incluso si dan soporte a datos de host big-endian. Al trabajar con Db2 o Pro *COBOL, debe utilizar el formato binario nativo predeterminado (BINARY (NATIVE) o BINARY (LE)).
- Puesto que IBM MQ espera un formato binario diferente que IBM Db2 y Oracle Pro *COBOL, no se recomienda tener llamadas MQ y SQL en la misma unidad de compilación o compilación por lotes.

## Datos nacionales

Los compiladores IBM COBOL utilizan la representación nativa UTF-16 de la plataforma al manejar datos nacionales.

En Linux x86, los elementos de datos nacionales se almacenarán y manipularán en formato UTF-16 little-endian. En AIX, los elementos de datos nacionales se almacenarán y manipularán en formato UTF-16 big-endian.

Al migrar aplicaciones COBOL de COBOL para AIX a COBOL for Linux en x86, es posible que obtenga resultados inesperados si el programa accede a datos almacenados en formato big-endian como compilador y el tiempo de ejecución tratará los datos como formato little-endian de forma predeterminada..

Puede utilizar la opción UTF16(BE) para informar al compilador COBOL for Linux en x86 de que maneje elementos de datos nacionales en formato big-endian coherentes con COBOL for AIX; sin embargo, esto tendrá cierta sobrecarga de rendimiento ya que el compilador necesita convertir los datos a y desde su formato nativo, little-endian. Si un elemento de datos determinado necesita permanecer en la

representación little-endian (LE) cuando se ha especificado UTF16(BE), utilice la cláusula NATIVE en la sentencia USAGE.

**Tareas relacionadas**
"Establecimiento del entorno local" en *Guía de programación*
"Arreglo de diferencias causadas por representaciones de datos" en la publicación *Programming Guide*

**Referencias relacionadas**
"CHAR" en la *Guía de programación*
"SOSI" en la *Guía de programación*

## Variables de entorno de compilador y tiempo de ejecución

COBOL for Linux en x86 no reconoce las variables de entorno de tiempo de ejecución y compilador siguientes que están disponibles en COBOL para AIX.

- CLASSPATH
- COBJVMINITOPCIONES
- VOLCADO
- COBMSGS
- LIBPATH
- VÍA_ACCESO_UBICACIÓN
- TMP

Las variables de entorno relacionadas con ENCINA SFS disponibles con COBOL for AIX se han renombrado a CICS variables de entorno SFS en COBOL for Linux en x86.

COBOL for Linux en x86 reconoce varias variables de entorno de tiempo de ejecución del compilador y que no se utilizan en COBOL for AIX, tal como se indica a continuación.

- COBCORE
- COBOUTDIR
- DBCS_CODEPAGE
- LD_LIBRARY_PATH
- TMPDIR
- BASEDATOS_MONGOBD_VF
- URI de VFS_MONGODB_URI

Información relacionada:

"Establecimiento de variables de entorno" en *Guía de programación*

## Elementos de lenguaje

La tabla siguiente lista los elementos de lenguaje que son diferentes entre los compiladores COBOL for AIX y COBOL for Linux en x86 , y donde sea posible ofrece consejos sobre cómo manejar estas diferencias en los programas COBOL for Linux en x86 .

*Tabla 8.* ***Diferencias de lenguaje entre COBOL for AIX y COBOL for Linux en x86***

| **Elemento de idioma** | **Implementación o restricción de COBOL for Linux en x86** |
|---|---|
| Sentencia INVOKE | COBOL for Linux en x86 no da soporte a la escritura de programas orientados a objetos, a la creación de instancias de objeto de una clase COBOL o Java™ o a la invocación de un método definido en una clase COBOL o Java . |
| OBJECT REFERENCE elementos de datos | OBJECT REFERENCE no están soportados en COBOL for Linux en x86. |

| Tabla 8. ***Diferencias de lenguaje entre COBOL for AIX y COBOL for Linux en x86*** *(continuación)* | |
|---|---|
| **Elemento de idioma** | **Implementación o restricción de COBOL for Linux en x86** |
| Sistema de archivos SDU | COBOL for Linux en x86 no da soporte al sistema de archivos SDU. Si especifica SDU, se tratará como si se hubiera especificado STL. Cuando especifique VSAM como sistema de archivos, el valor predeterminado será STL. Consulte "Identificación de archivos" en la publicación *Programming Guide*. |

# Especificación de archivo

Existen algunas diferencias entre la forma en que COBOL for Linux en x86 maneja los archivos y la forma en que COBOL for AIX maneja los archivos.

## Apertura de archivos para grabación

La apertura de un archivo para su grabación es una llamada de no bloqueo en AIX, pero es una llamada de bloqueo en Linux. En AIX, si varios procesos están accediendo al mismo archivo, el proceso B no esperará a que el proceso A se realice con el archivo antes de que el proceso B abra el archivo para una grabación. Depende del usuario asegurarse de que sólo un proceso escribe en el archivo a la vez. En Linux, si un proceso ha abierto un archivo para una grabación, el segundo proceso esperará hasta que finalice el primer proceso para evitar que se dañe el archivo. Puede utilizar la variable de entorno VFS_LOCK=0 para cambiar el archivo abierto para la operación de escritura en Linux de nuevo a una llamada no de bloqueo. Debe utilizar esta variable de entorno con precaución, ya que existe la posibilidad de que se dañe el archivo.

# Avisos

Interfaces de programación: las interfaces de programación con intención permiten al cliente escribir programas para obtener los servicios de IBM COBOL for Linux en x86.

Esta información se ha desarrollado para productos y servicios ofrecidos en los Estados Unidos.

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 suministro de este documento no proporciona ninguna licencia sobre estas patentes. Puede enviar consultas sobre licencia, por escrito, a:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
U.S.A.

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:

Intellectual Property Licensing
Ley de propiedad intelectual y legal
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokio 103-8510, Japón

**El párrafo que sigue no se aplica al Reino Unido ni a ningún otro país en el que tales disposiciones sean incompatibles con la legislación local:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL", SIN GARANTÍA DE NINGUNA CLASE, YA SEA EXPLÍCITA O IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN, DE COMERCIALIZACIÓN O IDONEIDAD PARA UN PROPÓSITO 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 podría 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.

Cualquier referencia de esta información a sitios web que no sean de IBM se proporciona únicamente como ayuda y no se consideran en modo alguno como aprobados por IBM. Los materiales de dichos sitios web no forman parte del material de este producto de IBM y el usuario es el único responsable del uso que haga ellos.

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:

Departamento de propiedad intelectual para software Rational
IBM Corporation
5 Technology Park Drive
Westford, MA 01886
U.S.A.

Esta información estará disponible, bajo las condiciones adecuadas, incluyendo en algunos casos el pago de una cuota.

IBM proporciona el programa bajo licencia descrito en este documento y todo el material bajo licencia disponible para el mismo mediante los términos del Acuerdo de Cliente de IBM, el Acuerdo de Licencia de Programa Internacional de IBM o cualquier acuerdo equivalente entre las partes.

Los datos de rendimiento contenidos en este documento se han determinado en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar considerablemente. Algunas mediciones pueden haberse efectuado en sistemas a nivel de desarrollo, y no es seguro que esas mediciones sean las mismas en sistemas disponibles a nivel 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 verificar los datos aplicables 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 manera más completa posible, los ejemplos incluyen los nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con nombres y direcciones utilizados por una empresa real es totalmente casual.

**LICENCIA DE DERECHOS DE AUTOR:**

Esta información contiene programas de aplicación de ejemplo en lenguaje fuente, que ilustran las técnicas de programación en diversas plataformas operativas. El cliente puede copiar, modificar y distribuir estos programas de ejemplo de cualquier modo sin efectuar ningún pago a IBM con el propósito de desarrollar, utilizar, comercializar o distribuir programas de aplicación en conformidad con la interfaz de programación de aplicaciones de la plataforma operativa para la que se hayan escrito dichos programas de ejemplo. Estos ejemplos no se han verificado exhaustivamente bajo todas las condiciones. Por lo tanto, IBM no puede garantizar ni implicar la fiabilidad, la capacidad de servicio ni el funcionamiento de estos programas. Los programas de ejemplo se proporcionan "TAL CUAL", sin garantía de ningún tipo. IBM no se responsabilizará de los daños causados por el uso de los programas de ejemplo.

Cada copia o cualquier porción de estos programas de ejemplo o cualquier trabajo derivativo debe incluir un aviso de copyright, tal como se indica a continuación:

© (nombre de la empresa) (año). Las partes de este código se derivan de IBM Corp. Programas de ejemplo. © Copyright IBM Corp. 2010, 2015.

**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 las 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 ni otras tecnologías para recopilar información de identificación personal.

Si las configuraciones desplegadas para esta oferta de software le ofrecen como cliente la posibilidad de recopilar información de identificación personal de los usuarios finales mediante cookies y otras tecnologías, debe buscar asesoramiento legal sobre las leyes aplicables 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 IBMen http://www.ibm.com/privacy y IBMen http://www.ibm.com/privacy/details en la sección titulada "Cookies, Web Beacons and Other Technologies," y la "Declaración de privacidad deIBM Software Products and Software-as-a-Service" en http://www.ibm.com/software/info/product-privacy.

## Marcas registradas

IBM, el logotipo de IBM e ibm.com son marcas registradas de International Business Machines Corp. registradas en numerosas jurisdicciones de todo el mundo. Otros productos y nombres de servicios pueden ser marcas registradas de IBM o de otras empresas. Puede obtener una lista actualizada de marcas registradas de IBM en la web en "Información sobre copyright y marcas registradas" en www.ibm.com/legal/copytrade.shtml.

Número de Programa: 5737-L11

SC28-3454-00