PORTADA Tapa del Libro

AS/400 Advanced Series

CL Programación

Versión 3 Release 6

Número de Documento SC10-9637-00

AVISOS Avisos

¡Atención!

Antes de utilizar esta información y el producto al que da soporte, asegúrese de leer la información general que se encuentra en el apartado "Avisos" en el tema PORTADA_1.

VERSION Nota de Versión
Primera Edición (Diciembre 1995)

Esta publicación es la traducción del original en inglés AS/400 Advanced Series CL Programming, SC41-4721-00.

Esta edición se aplica al programa bajo licencia IBM Operating System/400, (Programa 5716-SS1), Versión 3 Release 6 Modificación 0, y a todos los releases y modificaciones posteriores hasta que se indique lo contrario en nuevas ediciones. Asegúrese de utilizar la edición adecuada al nivel del producto.

Efectúe el pedido de publicaciones al representante de IBM o a la sucursal de IBM de su localidad. Si vive en los Estados Unidos, Puerto Rico o Guam, puede solicitar las publicaciones a IBM Software Manufacturing Solutions, 800+879-2755. En la dirección que figura más abajo no hay existencias de publicaciones.

Al final de esta publicación encontrará una hoja para los comentarios del lector. Si falta dicho formulario, puede dirigir sus comentarios a:

|IBM S.A.
|National Language Solutions Center
|Avda. Diagonal 571
|08029 Barcelona
|España

Si lo prefiere, también puede enviar sus comentarios por fax dirigiéndose a:

Desde España: (93) 321 61 34
Desde otros países: 34 3 321 31 34

Si tiene acceso a Internet, puede enviar sus comentarios electrónicamente a *PUBAS400@VNET.IBM.COM*.

Cuando envía información a IBM, otorga a IBM un derecho no exclusivo a utilizar o distribuir dicha información de la manera que crea adecuada sin incurrir por ello en ninguna obligación ni restringirle su utilización.

CONTENIDO Contenido

PORTADA_1 Avisos

Las referencias que se hacen en la presente publicación a productos, programas o servicios de IBM no implican necesariamente que IBM tenga la intención de ponerlos a disposición del público en todos los países en los que IBM opera. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo puede utilizarse dicho producto, programa o servicio. Un producto, programa o servicio funcionalmente equivalente que no infrinja ninguno de los derechos de propiedad intelectual de IBM puede utilizarse en lugar del producto IBM. Se entiende que la evaluación y verificación de operaciones conjuntas con otros productos, excepto aquellos que designe expresamente IBM, son responsabilidad del usuario.

IBM puede tener patentes o solicitudes de patentes pendientes que cubren el tema tratado en este documento. La adquisición de este manual no le otorga ninguna licencia sobre dichas patentes. Puede enviar consultas sobre licencias por escrito a IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594, U.S.A.

|Los poseedores de licencias de este programa que deseen información sobre
|el mismo, con el propósito de habilitar: (i) el intercambio de información
|entre programas creados independientemente y otros programas (incluyendo
|este mismo) y (ii) el uso mutuo de la información que ha sido
|intercambiada, deben contactar con el coordinador de interoperatividad de
|software. Es posible que esta información esté disponible, sujeta a
|determinados términos y condiciones, incluyendo en algunos casos, el pago
|de una tarifa.

|Dirija sus preguntas a :

|IBM Corporation
|Software Interoperability Coordinator
|3605 Highway 52 N
|Rochester, MN 55901-9986 USA

Esta publicación puede contener imprecisiones técnicas o errores tipográficos.

Es posible que la presente publicación haga mención de productos que se han anunciado pero que no se encuentran actualmente disponibles en su país. Igualmente, también puede hacer referencia a productos que no se hayan hayan anunciado en su país. IBM no se compromete a proveer ninguno de los productos que se citan en el texto. La decisión final de anunciar un producto depende exclusivamente del juicio técnico y comercial de IBM.

Los cambios o adiciones que ha sufrido el texto vienen indicados por una línea vertical (|) a la izquierda de la modificación.

Consulte la sección "Resumen de Modificaciones" en el tema PORTADA_2 para obtener un resumen de los cambios realizados en el manual *CL Programación* y del modo en que están descritos en esta publicación.

Esta publicación contiene ejemplos de datos e informes que se utilizan cotidianamente en las operaciones comerciales. En un intento de ilustrarlos de la mejor manera posible, dichos ejemplos hacen mención de individuos, compañías, marcas y productos. Todos esos nombres son ficticios y cualquier parecido con los nombres y direcciones que utilicen empresas reales es pura coincidencia.

Esta publicación contiene pequeños programas facilitados por IBM a modo de ejemplo. IBM no ha comprobado completamente dichos ejemplos bajo todas las condiciones posibles, por lo que no puede garantizar ni implicar la fiabilidad, utilidad y el funcionamiento de dichos programas. Todos ellos se proporcionan "TAL CUAL" en este manual. QUEDA EXPRESAMENTE DENEGADA LA GARANTÍA IMPLÍCITA DE COMERCIALIZACIÓN Y ADECUACIÓN A UN FIN PARTICULAR.

Si está viendo este manual en copia software, las ilustraciones gráficas no aparecen.

Subtemas
PORTADA_1.1 Información sobre las Interfaces de Programación
PORTADA_1.2 Marcas registradas

PORTADA_1.1 Información sobre las Interfaces de Programación

|El manual *CL Programación* está destinado a ayudar al programador de aplicación. Esta publicación documenta la Interfaz de Programación de Uso General e Información de Ayuda Asociada proporcionada por el programa bajo licencia OS/400.

Las interfaces de programación de uso general permiten al usuario escribir programas que obtienen los servicios del programa bajo licencia OS/400.

|*PORTADA_1.2 Marcas registradas*

Los siguientes términos son marcas registradas de IBM Corporation en Estados Unidos y/o en otros países:

Application System/400	Integrated Language Environment
AS/400	Operating System/400
C/400	Operational Assistant
COBOL/400	OS/400
GDDM	RPG IV
IBM	400

|Otros nombres de compañías, productos o servicios, marcados con un doble |asterisco (**), pueden ser marcas registradas o marcas de servicio de |terceros.

PREFACIO Acerca de CL Programación (SC10-9637 (SC41-4721))
Este manual proporciona una amplia explicación de los temas de programación AS/400, incluyendo:

- Programación en lenguaje de control
- Conceptos de programación del AS/400
- Objetos y bibliotecas
- Manejo de mensajes
- Mandatos definidos por el usuario
- Menús definidos por el usuario
- Funciones de prueba

Subtemas
PREFACIO.1 Quién Debe Utilizar este Libro

PREFACIO.1 Quién Debe Utilizar este Libro

Esta guía va destinada al programador AS/400 o programador de aplicaciones, incluyendo programadores no de CL. Aunque la programación CL se trata en detalle, la mayor parte del material de esta guía se aplica al sistema en general y puede ser usada por todos los programadores de lenguajes de alto nivel soportados por el sistema AS/400.

Antes de utilizar este libro, debe estar familiarizado con los conceptos y la terminología general de programación, así como tener una visión general del Operating System/400 (OS/400) y del sistema AS/400. También debe estar familiarizado con las estaciones de trabajo que utilice.

Tal vez tenga que consultar otros manuales de IBM para obtener información
|más detallada acerca de un tema en particular. La publicación *Manual de*
|*consulta de publicaciones* proporciona información sobre todos los manuales de la biblioteca AS/400.

|Para mas información acerca de otras publicaciones AS/400, consulte:

|□ La publicación *Manual de consulta de publicaciones*, SC10-9613
| (SC41-4003), en la biblioteca en soporte software AS/400.
|□ *AS/400 Consulta de publicaciones*, una exclusiva interfaz multimedia
| para base de datos de búsqueda que contiene descripciones de títulos
| disponibles en IBM o en otros editores. *AS/400 Information Directory*
| se envía con su sistema sin ningún tipo de cargo.

Para ver una lista de las publicaciones relacionadas, véase la Bibliografía.

|*PORTADA_2 Resumen de Modificaciones*

Subtemas
PORTADA_2.1 Cambios Diversos

PORTADA_2.1 Cambios Diversos

Existe información nueva y cambiada, aunque no exclusivamente, en los siguientes capítulos y apéndices:

- Capítulo 2, "Programación CL" en el tema 2.0

  Consulte "Valores que Pueden Utilizarse como Variables" en el tema 2.6 para obtener información acerca de las dos siguientes API nuevas.

  - Obtener Hora Local Actual (CEELOCT)
  - Obtener Fecha Local Actual (CEEDATE)

- Capítulo 8, "Trabajar con Mensajes" en el tema 8.0

  Consulte "Cola de Mensajes QSYSMSG" en el tema 8.5 para obtener descripciones acerca de los siguientes mensajes de cola QSYSMSG nuevos:

  - CPI1159
  - CPI1160
  - CPI1161
  - CPI1162
  - CPI1165
  - CPI1166
  - CPI1167
  - CPI1168
  - CPI1169

  Los siguientes mensajes, la mayoría para protección por duplicación de disco, ya no se soportan:

  - CPI0920
  - CPI0945
  - CPI0946
  - CPI0947
  - CPI0956
  - CPI0957
  - CPI0958
  - CPI0959
  - CPI0970
  - CPI0992
  - CPI0996
  - CPI0997

  El mensaje CPI1136 se continúa enviando cada hora, pero sólo si la protección por duplicación de disco está suspendida en uno o más discos.

- Capítulo 9, "Definición de Mandatos" en el tema 9.0

  Consulte "Restricciones de Parámetros y Tipos de Datos" en el tema 9.3 para obtener información acerca del parámetro Lista en Deplazamientos de Lista (LISTDSPL). LISTDSPL es un nuevo parámetro para el Parámetro sentencia de definición de mandato (PARM).

- Capítulo 10, "Depuración de Programas ILE" en el tema 10.0

  Consulte el capítulo 10 para información cambiada, además de la nueva información acerca de:

  - SET (Nuevo mandato de depuración ILE.)
  - Nuevas teclas PF

- Apéndice C, "Archivos de Salida de Anotaciones de Trabajo" en el tema C.0

  Consulte el "Apéndice C" para obtener información acerca de los siguientes campos nuevos en las anotaciones de trabajo primarias:

  - QMHCSP
  - QMHCRP
  - QMHLSP

  El "Apéndice C" también contiene algunos cambios en los campos ya existentes en las anotaciones de trabajo primarias.

1.0 Capítulo 1. Introducción

En esta introducción se describen diversos conceptos principales del Operating System/400* (OS/400*). Estos conceptos se explican con todo detalle en los capítulos siguientes.

La operación del sistema se controla mediante:

- Mandatos CL. Los mandatos CL se utilizan individualmente en trabajos por lotes e interactivos, (tales como desde la pantalla de Entrada de Mandatos) y en programas y procedimientos CL.

- Opciones de menú. La operación del sistema puede controlarse seleccionando opciones de menú. Los usuarios interactivos pueden utilizar los menús de AS/400* para realizar un gran número de tareas del sistema.

- Mensajes del sistema. Los mensajes del sistema se utilizan para la comunicación entre programas y procedimientos y para la comunicación entre programas, procedimientos y usuarios. Los mensajes pueden informar acerca de las condiciones de error y de la información de estado.

Subtemas

1.1 Lenguaje de Control

El **lenguaje de control (CL)** es la interfaz primaria para el sistema operativo y pueden utilizarla al mismo tiempo usuarios en distintas estaciones de trabajo. Una sola sentencia del lenguaje de control se denomina **mandato**. Los mandatos se pueden entrar de las siguientes formas:

- Individualmente desde una estación de trabajo.
- Como parte de trabajos por lotes.
- Como sentencias fuente para crear un programa o procedimiento CL.

Los mandatos se pueden entrar de uno en uno desde una línea de mandatos o desde la pantalla Entrada de Mandatos.

Para simplificar la utilización de CL, todos los mandatos utilizan una sintaxis coherente. Además, el sistema operativo proporciona soporte de solicitud para todos los mandatos, valores por omisión para la mayoría de parámetros de mandatos y comprobación de validez para asegurar que un mandato se entra correctamente antes de realizar la función. De este modo, CL facilita una interfaz única y flexible para numerosas funciones distintas del sistema que pueden utilizar usuarios de diferentes sistemas.

Subtemas

1.1.1 Procedimiento

Un **Procedimiento** es un conjunto de sentencias en lenguaje de alto nivel que ejecuta una determinada tarea y después vuelve al programa que lo llamó.

En CL, un procedimiento suele empezar por una sentencia PGM y terminar por una sentencia ENDPGM.

1.1.2 Módulo

Un **Módulo** es el objeto resultante de la compilación del fuente. Un módulo debe estar enlazado a un programa para que funcione.

Un módulo CL consta del procedimiento escrito por el usuario y de un procedimiento de entrada de programa generado por el compilador CL. En otros HLL (por ejemplo, C), un módulo puede contener más de un procedimiento.

1.1.3 Programa

Un **Programa** ILE es un objeto OS/400 que combina uno o más módulos. Los módulos no pueden ejecutarse hasta que están enlazados a programas. Un programa debe tener un procedimiento de entrada de programa. El compilador CL genera un procedimiento de entrada de programa en cada módulo que crea. Un programa CL OPM es el objeto resultante de la compilación del fuente utilizando el mandato CRTCLPGM.

1.1.4 Programa de Servicio

Un **Programa de servicio** es el objeto OS/400 que combina uno o más módulos. Los programas de servicio no pueden llamarse directamente. Los procedimientos de un programa de servicio pueden llamarse desde otros procedimientos de programas y programas de servicio.

1.1.5 Sintaxis de Mandatos

Cada mandato consta de un nombre de mandato y parámetros. Un nombre de mandato está formado normalmente por un verbo, o acción, seguido de un nombre o frase que identifica el receptor de la acción. Las palabras que componen el nombre del mandato están abreviadas, normalmente con tres letras, para reducir la longitud de lo que se teclea cuando se entra el mandato. Por ejemplo, uno de los mandatos CL es Enviar Mensaje. El nombre del mandato es SNDMSG, que se utiliza para enviar un mensaje de un usuario a la cola de mensajes.

Los parámetros utilizados en los mandatos CL son los parámetros de palabra clave. La palabra clave, generalmente abreviada igual que los mandatos, identifica el propósito del parámetro. Sin embargo, cuando se entran mandatos, algunas palabras clave pueden omitirse especificando los parámetros en un cierto orden (especificación posicional).

1.2 Procedimientos CL

Los programas y procedimientos CL están formados por mandatos CL. Los mandatos se compilan en un programa OPM o un módulo que puede enlazarse a programas compuestos de módulos escritos en CL u otros lenguajes. Las ventajas de utilizar programas y procedimientos CL son las siguientes:

- La utilización de programas y procedimientos CL es más rápida que entrar y ejecutar los mandatos individualmente, ya que los mandatos se compilan y almacenan de forma que pueden ejecutarse.
- Los programas y procedimientos CL proporcionan un proceso consistente del conjunto de mandatos y de la lógica.
- Algunas funciones requieren mandatos CL que no se pueden entrar individualmente y deben formar parte de un programa o procedimiento CL.
- Los programas y procedimientos CL pueden probarse y depurarse como los demás programas y procedimientos en lenguaje de alto nivel (HLL).
- Se pueden pasar parámetros a programas y procedimientos CL para adaptar las operaciones ejecutadas por el programa o procedimiento a los requisitos concretos de esta utilización.
- Los módulos CL pueden enlazarse con otros procedimientos o módulos en lenguaje de alto nivel ILE* en un programa.

Los programas y procedimientos CL pueden utilizarse para muchos tipos de aplicaciones. Por ejemplo, los procedimientos pueden utilizarse para:

- Proporcionar una interfaz al usuario de una aplicación interactiva por medio de la cual el usuario puede solicitar funciones de aplicación sin necesidad de entender los mandatos utilizados en el programa o procedimiento. Esto facilita el trabajo del usuario de la estación de trabajo y reduce las posibilidades de errores producidos al entrar los mandatos.
- Controlar la operación de una aplicación estableciendo las variables utilizadas en la aplicación (como fecha, hora e indicadores externos) y especificando la lista de bibliotecas que utiliza la aplicación. Ello garantiza que estas operaciones se lleven a cabo cada vez que se ejecuta la aplicación.
- Proporcionar rutinas predefinidas para el operador del sistema, tales como procedimientos para arrancar un subsistema, proporcionar copias de seguridad de archivos o realizar otras funciones operativas. La utilización de programas y procedimientos CL reduce el número de mandatos que utiliza regularmente el operador, y garantiza que las operaciones del sistema se ejecutan de forma coherente.

La mayoría de los mandatos CL que proporciona el sistema se pueden utilizar en programas y procedimientos CL. Algunos mandatos están designados específicamente para utilizarlos en programas y procedimientos CL y no están disponibles cuando se entran mandatos individualmente. Estos mandatos incluyen:

- Mandatos de control lógico que pueden utilizarse para controlar qué operaciones realiza el programa o procedimiento según las condiciones existentes al ejecutarse el programa o procedimiento. Por ejemplo, *si (if)* existe una determinada condición, *entonces (then do)* efectuar ciertos procesos, *de lo contrario (else)* realizar otras operaciones. Estas operaciones lógicas proporcionan bifurcaciones condicionales e incondicionales en el programa o procedimiento CL.
- Operaciones de datos que posibilitan que el programa o procedimiento comunique con un usuario de estación de trabajo. Las operaciones de datos permiten al programa o procedimiento enviar datos con formato y recibir datos de la estación de trabajo, y permiten el acceso limitado a la base de datos.
- Mandatos que permiten al programa o procedimiento enviar mensajes al usuario de estación de pantalla.
- Mandatos que reciben mensajes enviados por otros programas y procedimientos. Estos mensajes pueden proporcionar una comunicación normal entre programas y procedimientos o indicar que existen errores u otras condiciones excepcionales.
- La utilización de variables y parámetros para pasar información entre mandatos del programa o procedimiento y entre programas y procedimientos.

- Llamar a otros procedimientos (sólo los programas pueden llamarse desde la línea de mandatos o en la corriente de trabajos por lotes).

Mediante la utilización de programas y procedimientos CL, pueden diseñarse aplicaciones con un programa o procedimiento independiente para cada función y con un programa o procedimiento CL que controla qué programas o procedimientos se ejecutan dentro de la aplicación. La aplicación puede constar de programas y procedimientos CL y otros programas o procedimientos HLL. En este tipo de aplicaciones, los programas y procedimientos CL se utilizan para:

- Determinar qué programas o procedimientos de la aplicación van a ejecutarse.
- Proporcionar funciones del sistema que no están disponibles a través de otros lenguajes HLL.
- Proporcionar interacción con el usuario de aplicación.

Los programas y procedimientos CL proporcionan la flexibilidad necesaria para que el usuario de la aplicación seleccione qué operaciones se deben efectuar y para que se ejecuten los procedimientos necesarios.

1.3 Definición de Mandatos

La **definición de mandatos** permite a los usuarios crear mandatos adicionales para satisfacer las necesidades específicas de una aplicación. Estos mandatos son similares a los mandatos del sistema.

Cada mandato del sistema tiene un objeto de definición de mandato y un programa de proceso de mandatos (CPP). El objeto de definición de mandato define el mandato, incluyendo:

- El nombre del mandato
- El CPP
- Los parámetros y valores que son válidos para el mandato
- Información de comprobación de validez que el sistema puede utilizar para validar el mandato cuando éste se entra
- Texto de solicitud que se visualiza si se pide una solicitud para el mandato.
- Información de ayuda en línea

El **CPP** es el programa al que se llama cuando se entra el mandato. Debido a que el sistema realiza la comprobación de validez cuando se entra el mandato, el CPP no siempre tiene que comprobar los parámetros que se le pasan.

Las funciones de definición de mandatos pueden utilizarse para:

- Crear mandatos exclusivos que los usuarios del sistema necesitan mientras se mantiene una interfaz coherente para los usuarios del mandato CL.

- Definir versiones alternativas de los mandatos CL para satisfacer los requisitos de los usuarios del sistema. Esta función puede implicar que se tengan diferentes valores por omisión para los parámetros o que se simplifiquen los mandatos de modo que algunos parámetros no necesiten entrarse. Pueden definirse valores constantes para dichos parámetros. Los mandatos proporcionados por IBM* no deben modificarse.

Véase el Capítulo 9, "Definición de Mandatos", para obtener una información más detallada sobre la definición de mandatos.

1.4 Menús

El sistema proporciona un extenso número de menús que permite a los usuarios realizar muchas de las funciones simplemente seleccionando opciones de menú. Las ventajas de usar menús para realizar las tareas del sistema son las siguientes:

- Los usuarios no necesitan comprender los mandatos CL ni la sintaxis de los mismos.
- El número de teclas a pulsar y la posibilidad de errores se reducen considerablemente.

La publicación *Operación del Sistema para Nuevos Usuarios* proporciona descripciones detalladas de menús del sistema e información referente a cómo utilizar estos menús.

La información referente a la creación de menús que pueden utilizarse como menús suministrados por el sistema se describe en la publicación Application Display Programming.

1.5 Objetos y Bibliotecas

Un **objeto** es un espacio de almacenamiento que tiene un nombre y que está formado por un conjunto de características que lo describen y, en algunos casos, por datos. Un objeto es cualquier cosa que existe y ocupa espacio en el almacenamiento y sobre el que se pueden realizar operaciones. Los atributos de un objeto incluyen el nombre, tipo, tamaño, la fecha de creación, y una descripción proporcionada por el usuario que ha creado el objeto. El valor de un objeto es la información almacenada en el objeto. El valor de un programa, por ejemplo, es el código que compone el programa. El valor de un archivo es el grupo de registros que componen el archivo. El concepto de objeto simplemente proporciona un término que puede usarse para referirse a un número de diferentes elementos que pueden almacenarse en el sistema, sin tener en cuenta qué son.

Subtemas

1.5.1 Objetos

Las funciones realizadas por la mayoría de los mandatos CL se aplican a los objetos. Algunos mandatos pueden utilizarse en cualquier tipo de objeto y otros se aplican solamente a un tipo específico de objeto.

El sistema soporta diversos tipos de objetos exclusivos. Algunos tipos identifican objetos comunes a muchos sistemas de proceso de datos, como son:

- Archivos
- Programas
- Mandatos
- Bibliotecas
- Colas
- Módulos
- Programas de servicio

Otros tipos de objetos son menos corrientes, tales como:

- Perfiles de usuario
- Descripciones de trabajo
- Descripciones de subsistema
- Descripciones de dispositivo

Los distintos tipos de objetos tienen características operativas diferentes. Estas diferencias hacen que cada tipo de objeto sea único. Por ejemplo, puesto que un archivo es un objeto que contiene datos, sus características operativas difieren de las de un programa que contiene instrucciones.

Cada objeto tiene un nombre. El nombre y el tipo de objeto se utilizan para identificar un objeto. El nombre de objeto lo asigna el usuario que crea el objeto. El tipo de objeto viene determinado por el mandato utilizado para crearlo. Por ejemplo, si se crea un programa con el nombre ACEP (para *actualización entrada de pedidos*), el programa siempre se podrá localizar mediante ese nombre. El sistema utiliza el nombre del objeto (ACEP) y el tipo de objeto (programa) para localizar el objeto y realizar las operaciones sobre él. Pueden existir objetos distintos con el mismo nombre, pero deberán ser de diferente tipo o estar almacenados en bibliotecas distintas.

El sistema mantiene su integridad evitando la mala utilización de ciertas funciones, dependiendo del tipo de objeto. Por ejemplo, el mandato CALL hace que se ejecute un objeto de programa. Si especificase CALL y nombrara un archivo, el mandato fallaría porque el tipo de objeto debe ser un programa.

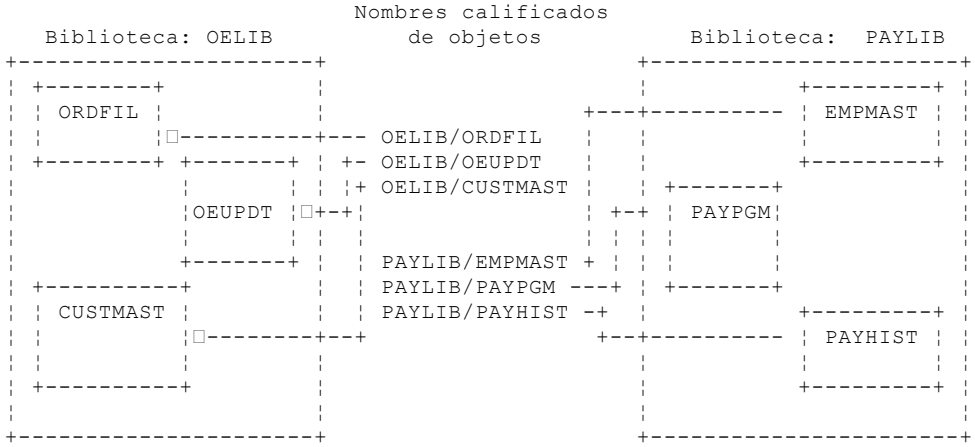
1.5.2 Bibliotecas

Una **biblioteca** es un objeto utilizado para agrupar objetos relacionados, y para buscar objetos por el nombre cuando se utilizan. Por lo tanto, una biblioteca es un directorio de un grupo de objetos. Las bibliotecas pueden utilizarse para agrupar objetos de cualquier modo que tenga sentido. Por ejemplo, los objetos que pueden agruparse de acuerdo con los requisitos de seguridad, de copia de seguridad o de proceso. El número de objetos que puede contener una biblioteca y el número de bibliotecas que puede tener el sistema sólo está limitado por la cantidad de almacenamiento disponible, pero debe ser menor de 8000 para garantizar que se podrán realizar las operaciones de salvar.

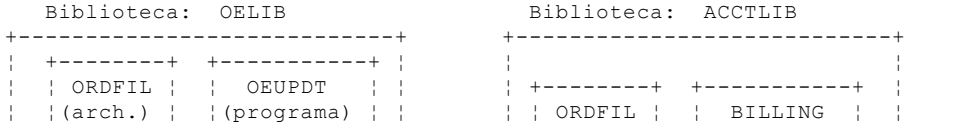
La agrupación de objetos efectuada por las bibliotecas es una agrupación lógica. Cuando se crea una biblioteca, puede especificar en qué agrupación de almacenamiento auxiliar (ASP) debe crearse. Todos los objetos creados en una biblioteca se crean en la misma ASP que ésta. Los objetos de una biblioteca no se encuentran, de forma obligatoria, físicamente uno junto a otro. El tamaño de una biblioteca o de cualquier otro objeto no está restringido por la cantidad de espacio adyacente disponible en el almacenamiento. El sistema busca el almacenamiento necesario para los objetos a medida que se almacenan en el sistema.

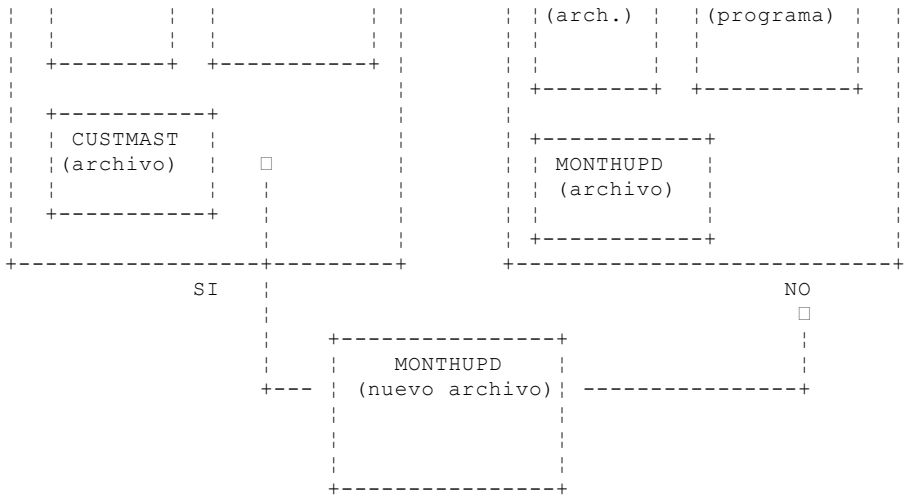
La mayoría de tipos de objetos se colocan en una biblioteca cuando se crean. Un objeto adopta la la autorización pública de la biblioteca en la que se coloca. Esta autorización quedó definida cuando se creó la biblioteca utilizando el parámetro CRTAUT en el mandato Crear Biblioteca (CRTLIB). La mayor parte de tipos de objeto se pueden mover de una biblioteca a otra, pero un objeto único no puede estar en más de una biblioteca al mismo tiempo. Cuando se traslada un objeto a otro biblioteca, el objeto no cambia de posición en el almacenamiento, pero se le localiza a través de la nueva biblioteca. La mayoría de tipos de objeto también pueden cambiarse de nombre o copiarse de una biblioteca a otra.

Un nombre de biblioteca puede utilizarse para proporcionar otro nivel de identificación al nombre de un objeto. Como se ha descrito anteriormente, un objeto se identifica por su nombre y su tipo. El nombre de la biblioteca califica más el nombre del objeto. La combinación de un nombre de objeto y el nombre de la biblioteca se denomina *nombre calificado* del objeto. El nombre calificado indica al sistema el nombre del objeto y la biblioteca en la que se encuentra. El siguiente diagrama muestra dos bibliotecas y los nombres calificados de los objetos contenidos en ellas:



Pueden existir dos objetos con el mismo nombre y tipo en bibliotecas distintas. Dos objetos distintos con el mismo nombre no pueden existir en la misma biblioteca a menos que sus tipos de objeto difieran. Este diseño permite a un programa hacer referencia a objetos por el nombre para trabajar con objetos distintos (objetos con el mismo nombre pero almacenados en bibliotecas distintas) en ejecuciones sucesivas del programa sin modificar el programa. Asimismo, un usuario de estación de trabajo que está creando un objeto nuevo no necesita preocuparse de los nombres utilizados para los objetos de otras bibliotecas. Por ejemplo, en el diagrama siguiente, un nuevo archivo denominado MONTHUPD (actualización mensual) podría añadirse a la biblioteca OELIB, pero no a la biblioteca ACCTLIB. La creación del archivo en ACCTLIB fallaría debido a que otro objeto denominado MONTHUPD con el mismo tipo de archivo ya existe en la biblioteca ACCTLIB.





Un objeto se identifica en una biblioteca por el nombre de objeto y por el tipo. Muchos mandatos CL se aplican solamente a un único tipo de objeto, de modo que no es necesario identificar explícitamente el tipo de objeto. En el caso de los mandatos que se aplican a varios tipos de objeto, debe indicarse explícitamente el tipo.

1.5.3 Utilización de Bibliotecas para Buscar Objetos

Un nombre de objeto puede especificarse como un nombre calificado (en el que se especifica el nombre de la biblioteca y el nombre del objeto) o puede especificarse sólo el nombre del objeto. Si se especifica un nombre calificado, el sistema intenta encontrar el objeto en la biblioteca indicada. Si sólo se especifica el nombre del objeto, el sistema busca en la *lista de bibliotecas*, hasta que encuentra la primera aparición del objeto con el nombre y el tipo correcto o hasta que ha buscado en todas las bibliotecas de la lista sin encontrar el objeto. Las bibliotecas en las que se busca y el orden de búsqueda se determinan mediante una lista de búsqueda llamada lista de bibliotecas. El sistema crea una lista de bibliotecas inicial para cada trabajo cuando éste se inicia.

Nota: Los valores iniciales de las partes del sistema y del usuario de la lista de bibliotecas se encuentran en los valores del sistema QSYSLIBL y QUSRLIBL. Estos valores del sistema pueden alterarse temporalmente en la descripción de trabajo.

Una lista de bibliotecas tiene cuatro partes. La primera es la *parte del sistema* de la lista de bibliotecas. Se busca en las bibliotecas de la parte del sistema de la lista antes de buscar en las demás bibliotecas de la lista. Esta parte especifica las bibliotecas utilizadas para todos los trabajos que se ejecutan en el sistema. Cuando se instala el sistema, la parte del sistema de la lista de bibliotecas consta de la biblioteca del sistema (QSYS), la biblioteca de datos del usuario del sistema (QUSRSYS), la biblioteca de información de ayuda del sistema (QHLPSYS) y la biblioteca de Interfaces de Programación de Comunicaciones (CPI) del sistema (QSYS2).

Nota: La biblioteca QSYS2 contiene objetos que no cumplen los convenios de denominación del OS/400*. Estos objetos *no* empiezan por la letra Q y por tanto pueden estar en conflicto con los nombres de objeto de usuario.

La segunda parte de la lista de bibliotecas es la parte de *biblioteca de productos*. Contiene de biblioteca de productos cuando los usuarios ejecutan mandatos o menús para incluir la biblioteca donde se almacenan los objetos de producto. La biblioteca de productos variará cuando se ejecuta un trabajo, según la función que se realice.

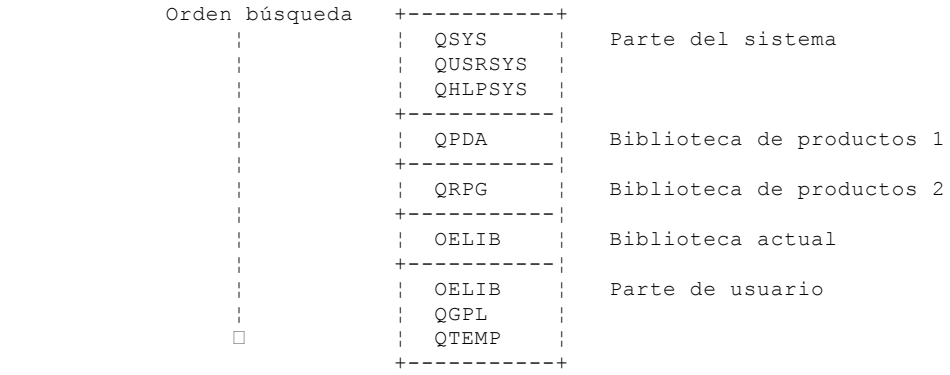
La tercera parte de la lista de bibliotecas es la *biblioteca actual*. Contiene es la biblioteca por omisión utilizada para crear objetos. Los usuarios pueden especificar la biblioteca actual para un trabajo utilizando mandatos CL.

La cuarta parte de la lista de bibliotecas es la *parte de usuario*. Contiene las bibliotecas utilizadas por los programas de aplicación para realizar sus funciones. Cuando se instala el sistema, esta parte contiene la biblioteca de uso general (QGPL) y la biblioteca temporal de trabajo (QTEMP). Cada trabajo tiene su propia QTEMP, que no es visible para los demás trabajos. QTEMP se borra cuando un trabajo termina.

Cuando un sistema tiene un número de bibliotecas definidas por el usuario, la parte de usuario de la lista de bibliotecas puede ser diferente para trabajos distintos. Por ejemplo, para un trabajo de entrada de pedidos, la parte de usuario de la lista de bibliotecas podría incluir:

- OELIB (biblioteca de entrada de pedidos)
- QGPL (biblioteca de uso general)
- QTEMP (biblioteca temporal de trabajo)

El siguiente diagrama muestra un ejemplo de una lista de bibliotecas:



La especificación del nombre de objeto solamente y el uso de la lista de bibliotecas para buscar el objeto puede hacer más fácil y flexible la utilización del sistema AS/400. Puede diseñarse una lista de bibliotecas para cada trabajo para garantizar que los objetos correctos se localizarán

sin utilizar los nombres calificados. Este planteamiento aporta ventajas tales como:

- Una más fácil comprobación de los programas de aplicación. Las bibliotecas pueden crearse para contener datos de muestra al someterse a prueba los programas. Los nombres de objetos utilizados en la biblioteca son los mismos que aquéllos utilizados en la biblioteca de producción normal. La biblioteca con los objetos de prueba se coloca antes de la biblioteca de producción normal en la lista de bibliotecas. Cuando el programa ha sido totalmente comprobado, esa biblioteca puede eliminarse de la lista de bibliotecas. El programa trabaja entonces con los objetos contenidos en las bibliotecas de producción y los nombres de objetos no necesitan cambiarse en el programa.

- Una utilización flexible de las bibliotecas en el sistema. De la misma forma que el proceso necesita cambiar, las bibliotecas existentes pueden necesitar dividirse en más de una para ayudar a simplificar la organización de los objetos en el sistema. Este cambio no exige que se cambien los nombres de los objetos en los programas. Solamente necesitan cambiarse las listas de bibliotecas utilizadas por los trabajos.

- La posibilidad de permitir que usuarios de diversos sistemas trabajen con objetos distintos utilizando el mismo programa de aplicación. Pueden crearse bibliotecas separadas para cada usuario o grupo de usuarios. La lista de bibliotecas para cada trabajo de usuario garantiza que el programa utilizará los objetos correctos para cada usuario del sistema.

- La posibilidad que tienen los mismos programas de aplicación de funcionar con datos diferentes. Por ejemplo, puede tener múltiples empresas o subconjuntos de una empresa que se procesan independientemente. El tener una biblioteca para cada empresa o subconjunto le permite guardar los datos separadamente.

Gracias a estas ventajas, los nombres calificados no suelen especificarse cuando se utilizan objetos ya existentes. Sin embargo, si decide *no* utilizar nombres calificados, tenga en cuenta la posible exposición de seguridad. Por ejemplo, si llama a un programa sin un nombre calificado, otro usuario podrá cambiar la lista de bibliotecas y llamar a una versión diferente del programa que hace mal uso de información confidencial.

1.6 Mensajes

Un **mensaje** es una comunicación enviada desde un usuario, programa o procedimiento a otro. La mayoría de sistemas de proceso de datos proporcionan comunicaciones entre el sistema y el operador, para manejar errores y otras situaciones que se producen durante el proceso. OS/400 también proporciona funciones de manejo de mensajes, que soportan comunicaciones a dos vías entre programas y usuarios de sistema, entre programas, entre procedimientos dentro de un programa y entre usuarios de sistema. Se soportan dos tipos de mensajes:

- Mensajes inmediatos, creados por el programa o el usuario del sistema cuando se envían y no se almacenan permanentemente en el sistema.

- Mensajes predefinidos, creados antes de utilizarlos. Estos mensajes se colocan en un archivo de mensajes cuando se crean y se recuperan de dicho archivo cuando se utilizan.

Debido a que los mensajes pueden utilizarse para proporcionar comunicaciones entre programas, entre procedimientos de un programa, o entre programas y usuarios, debe tenerse en cuenta la utilización de las funciones de manejo de mensajes OS/400 al desarrollar aplicaciones. Los conceptos siguientes de manejo de mensajes son importantes para el desarrollo de aplicaciones:

- Los mensajes pueden definirse en archivos de mensajes, que están fuera de los programas que los utilizan, y se puede proporcionar información variable en el texto de mensaje cuando éste se envía. Puesto que los mensajes se definen fuera de los programas, éstos no han de modificarse cuando se cambian los mensajes. Este enfoque también permite que el mismo programa se utilice con archivos de mensajes que contengan traducciones de los mensajes a diferentes idiomas.

- Los mensajes se envían a y se reciben desde las colas de mensajes, que son objetos separados en el sistema. Un mensaje enviado a una cola puede permanecer en la cola hasta que un programa o un usuario de la estación de trabajo lo reciba explícitamente.

- Un programa pueden enviar mensajes a un usuario que solicitó el programa sin tener en cuenta en que estación de trabajo ha iniciado la sesión ese usuario. Los mensajes no tienen que enviarse a un dispositivo específico; puede utilizarse un programa desde estaciones de trabajo distintas sin cambio alguno.

Consulte la publicación National Language Support para obtener información referente al Identificador de Juego de Caracteres (CCSID) para menús, mensajes y descripciones de mensaje.

Subtemas

1.6.1 Descripciones de Mensaje

Una **descripción de mensaje** define un mensaje al OS/400. La descripción de mensaje contiene el texto del mensaje e información sobre variables de sustitución, y puede incluir datos variables proporcionados por el emisor cuando se envía el mensaje.

Las descripciones de mensajes se almacenan en archivos de mensajes. Cada descripción debe tener un identificador que sea exclusivo en el archivo. Cuando se envía un mensaje, el archivo de mensajes y el identificador del mensaje indican al sistema qué descripción de mensaje debe utilizarse.

1.6.2 Colas de Mensajes

Cuando se envía un mensaje a un procedimiento, programa o usuario del sistema, se sitúa en una **cola de mensajes** asociada con ese procedimiento, programa o usuario. El procedimiento, programa o usuario ve el mensaje al recibirlo desde la cola.

El OS/400 proporciona colas de mensajes para:

- Cada estación de trabajo en el sistema
- Cada usuario incorporado al sistema
- El operador del sistema
- Las anotaciones históricas del sistema

Pueden crearse colas de mensajes adicionales para satisfacer algunos requisitos especiales de las aplicaciones. Los mensajes enviados a las colas de mensajes se guardan, por lo que no es necesario que el receptor del mensaje lo procese inmediatamente.

1.7 Funciones de Prueba

El sistema incluye funciones que permiten al programador observar las operaciones realizadas mientras se ejecuta el programa. Estas funciones pueden utilizarse para localizar operaciones que no se están ejecutando tal como se esperaba. Las funciones de prueba pueden utilizarse en trabajos interactivos o de proceso por lotes desde una estación de trabajo. En ambos casos, el programa observado ha de estar en el entorno de prueba, llamado *modalidad de depuración*.

Las **funciones de prueba** facilitan la búsqueda de errores que son difíciles de encontrar en las sentencias fuente del procedimiento. A menudo, un error se descubre solamente porque la salida producida no es como se esperaba. Para encontrar esos errores, el programador necesita poder detener el programa en un punto dado (llamado *punto de interrupción*) y examinar la información de las variables del programa para ver si es correcta. Puede que desee realizar cambios en estas variables antes de permitir que el programa siga ejecutándose.

No es necesario tener conocimientos de instrucciones de lenguaje máquina, así como tampoco es necesario incluir instrucciones especiales en el programa para utilizar las funciones de prueba. Las funciones de prueba OS/400 le permiten:

- Parar un programa en ejecución en un punto determinado en las sentencias fuente del programa.
- Visualizar información sobre las variables del procedimiento en cualquier punto en el que el programa puede detenerse. También puede cambiar la información de las variables antes de continuar el proceso del procedimiento.

Consulte el Capítulo 10, "Depuración de Programas ILE" en el tema 10.0, para obtener más información acerca de la depuración de programas ILE* (Entorno de Lenguajes Integrados) o Apéndice A, "Depuración de Programas OPM" en el tema A.0 para obtener más información referente a la depuración de programas OPM.

Consulte la guía ILE adecuada para obtener información de depuración con otros lenguajes ILE.

2.0 Capítulo 2. Programación CL

El capítulo 2 se centra en ILE en vez de en OPM. Por ello, se utiliza el término 'procedimiento' en lugar de 'programa', en este capítulo. Sin embargo, cuando se describen mandatos CL en general, puede que se utilice el término 'programa'.

Un **procedimiento CL** es un grupo de mandatos CL que indican al sistema dónde obtener las entradas, cómo procesarlas y dónde colocar el resultado. Se asigna un nombre al procedimiento mediante el cual otros procedimientos pueden llamarlo o puede enlazarse a un programa y llamarlo. Al igual que con otros tipos de procedimientos, debe entrar sentencias fuente de procedimiento CL, compilarlas y enlazarlas antes de poder ejecutar el procedimiento.

Cuando entra los mandatos CL individualmente (desde la pantalla Entrada de Mandatos, por ejemplo, o como mandatos individuales en una corriente de entrada), cada mandato se procesa por separado. Cuando entra mandatos CL como sentencias fuente para un procedimiento CL, el fuente se conserva para una modificación posterior si lo decide, y los mandatos se compilan en un módulo. Este módulo permanece como un objeto del sistema permanente que puede enlazarse a otros programas y ejecutarse. Así pues, el CL es, de hecho, un lenguaje de programación de alto nivel para funciones del sistema. Los procedimientos CL garantizan un proceso consistente de grupos de mandatos. Se pueden ejecutar funciones con un procedimiento CL que no pueden realizarse entrando mandatos individualmente; además, el programa o procedimiento CL proporciona un mayor rendimiento en tiempo de ejecución que el proceso de diversos mandatos por separado.

Los procedimientos CL pueden utilizarse en proceso por lotes o interactivo. Ciertos mandatos o funciones están restringidos tanto en trabajos interactivos como en trabajos de proceso por lotes.

Las sentencias fuente CL están formadas por mandatos CL. No todos los mandatos CL pueden utilizarse como sentencias fuente CL, y algunos de ellos pueden utilizarse sólo en procedimientos CL o programas OPM. Puede determinar las restricciones en el uso de los mandatos CL examinando la marca en el recuadro situado en el ángulo superior derecho del diagrama de sintaxis de un mandato en la publicación CL Reference, como se muestra con el mandato Programa (PGM):

```
                                                                  Pgm: B,I

                                                        (P)
>>--PGM-----------------------------------------------------------------><
         ¦        <----------------------+ (1)    ¦
         +-PARM(----&nombre variable CL--------)-+
```

Notas:
(1) Un máximo de 40 repeticiones

(P) Todos los parámetros anteriores a este punto pueden especificarse posicionalmente.

Pgm: B,I en el diagrama de sintaxis para el mandato PGM muestra que este mandato puede utilizarse en trabajos de proceso por lotes o interactivos, pero puede utilizarse sólo dentro de un programa o procedimiento CL.

Los mandatos que se pueden utilizar solamente como sentencias fuente en programas y procedimientos CL sólo tendrán **Pgm:** en el recuadro. Si el recuadro no contiene este indicador, el mandato no puede utilizarse como fuente para un programa o procedimiento CL. En el volumen 1 de la publicación CL Reference encontrará más información sobre cómo leer un diagrama de sintaxis.

Las sentencias fuente pueden entrarse en un miembro fuente de base de datos de forma interactiva desde una estación de trabajo o en una corriente de entrada de trabajos de proceso por lotes desde un dispositivo. Para crear un programa utilizando sentencias fuente CL, debe entrar las sentencias fuente en un miembro fuente de base de datos. Entonces podrá crear un programa ILE compilando el miembro fuente en un módulo y enlazando el módulo dentro de un objeto programa.

Los procedimientos CL pueden escribirse por muchos motivos, entre los cuales se encuentran:

- Para controlar la secuencia de proceso y de llamada de otros programas o procedimientos.
- Para visualizar un menú y ejecutar mandatos en función de las opciones seleccionadas desde ese menú. Esto facilita el trabajo del usuario de la estación de trabajo y reduce los errores.

- Para leer un archivo de base de datos.

- Para manejar las condiciones de error emitidas desde los mandatos, programas o procedimientos mediante la supervisión de mensajes específicos.

- Para controlar la operación de una aplicación mediante el establecimiento de variables utilizadas en la aplicación, tales como la fecha, la hora y los indicadores externos.

- Para proporcionar funciones predefinidas para el operador del sistema, como arrancar un subsistema o salvar archivos. Ello reduce el número de mandatos que el operador utiliza regularmente y garantiza que las operaciones del sistema se realizan de forma coherente.

La utilización de procedimientos CL para una aplicación presenta muchas ventajas. Por ejemplo:

- Puesto que los mandatos se almacenan en un formato que se puede procesar cuando se crea el programa, es más rápido utilizar programas CL que entrar y ejecutar los mandatos de forma individual.

- Los procedimientos CL son flexibles. Los parámetros pueden pasarse a los procedimientos CL para adaptar las operaciones realizadas por el procedimiento a las necesidades de una utilización concreta.

- Los procedimientos CL pueden probarse y depurarse igual que otros programas o procedimientos de lenguaje de alto nivel.

- Los programas y procedimientos CL pueden incorporar lógica condicional y funciones especiales no disponibles cuando los mandatos se entran individualmente.

- Los procedimientos CL pueden enlazarse con procedimientos de otros lenguajes.

No puede utilizar los procedimientos CL para:

- Añadir o actualizar registros en los archivos de bases de datos.
- Utilizar archivos de impresora o archivos ICF.
- Utilizar subarchivos en los archivos de pantalla.
- Utilizar los archivos de pantalla descritos en el programa.

Subtemas

2.1 Creación de un Programa CL

Todos los programas se crean por pasos:

1. Creación del fuente. Los procedimientos CL constan de mandatos CL. En la mayoría de los casos, las sentencias fuente se entran en un archivo de base de datos en la secuencia lógica determinada por el diseño de la aplicación.

2. Creación del módulo. Al utilizar el mandato Crear Módulo de Lenguaje de Control (CRTCLMOD), se utiliza este fuente para crear un objeto del sistema. El módulo creado puede enlazarse a programas. Un módulo CL contiene un procedimiento CL. Otros lenguajes HLL pueden contener múltiples procedimientos para cada módulo.

3. Creación del programa. Con el mandato Crear Programa (CRTPGM), se utiliza este módulo (junto con otros módulos y programas de servicio) para crear un programa.

Nota: Si desea crear un programa que conste sólo de un módulo CL, utilice el mandato Crear Programa CL Enlazado (CRTBNDCL), que combina los pasos 2 y 3.

Subtemas

2.1.1 Entrada Interactiva

El sistema AS/400 proporciona un gran número de menús y pantallas para ayudar al programador, que incluyen el Menú de Programador, la pantalla Entrada de Mandatos, pantallas de solicitud de mandatos y el Menú del Gestor para el Desarrollo de Programas (PDM). Si su sistema AS/400 utiliza las funciones de seguridad descritas en el manual Security - Reference, la posibilidad que tiene de utilizar estas pantallas viene controlada por la autorización que se otorgó a su perfil de usuario. Normalmente es el responsable de seguridad del sistema quien crea y mantiene los perfiles de usuario.

El método de entrada de fuente utilizado con mayor frecuencia es el Programa de Utilidad para Entrada del Fuente (SEU), que forma parte del programa bajo licencia Programa de Utilidad para Entrada del Fuente AS/400.

2.1.2 Entrada por Lotes

Puede crear un fuente CL, un módulo CL o un programa en una corriente de entrada por lotes desde disquete. El siguiente ejemplo muestra las partes básicas de la corriente de entrada desde una unidad de disquetes. La entrada se somete a una cola de trabajos mediante el mandatos Someter Trabajo de Disquete (SBMDKTJOB). La corriente de entrada ha de tener este formato:

```
// BCHJOB
CRTBNDCL PGM(QGPL/EDUPGM) SRCFILE(PERLIST)
// DATA FILE(PERLIST) FILETYPE(*SRC)
   .
   .              (Fuente Procedimiento CL)
   .
//
/*
// ENDINP
```

Esta corriente crea un programa CL desde el fuente incorporado. Si desea mantener el fuente en línea, puede utilizar el mandato Copiar Archivo (CPYF) para copiar el fuente en un archivo de base de datos. El programa CL podría entonces crearse utilizando el archivo de base de datos.

También puede crear un módulo CL directamente desde un fuente CL en un soporte de almacenamiento externo, como un disquete, utilizando un archivo de dispositivo suministrado por IBM. El archivo fuente de disquete suministrado por IBM es QDKTSRC (utilice QTAPSRC para cintas). Suponga, por ejemplo, que las sentencias fuente CL están en un archivo fuente en un disquete denominado PGMA.

El primer paso es identificar la ubicación del fuente en el disquete utilizando el mandato de alteración temporal siguiente con la alteración temporal del atributo LABEL.

```
OVRDKTF FILE(QDKTSRC) LABEL(PGMA)
```

Ahora puede considerar el archivo QDKTSRC como el archivo fuente en el mandato Crear Módulo CL (CRTCLMOD). Para crear el módulo CL basándose en la entrada de fuente del disquete, entre el siguiente mandato:

```
CRTCLMOD MODULE(QGPL/PGMA) SRCFILE(QDKTSRC)
```

Cuando se procesa el mandato CRTCLMOD, éste trata al archivo fuente QDKTSRC como cualquier archivo fuente de base de datos. Utilizando la alteración temporal, se localiza el fuente en el disquete. Se crea PGMA en QPGL y el fuente para aquel programa permanece en el disquete. Consulte la publicación *Gestión de datos* para obtener más información referente a archivos de dispositivo.

2.1.3 Partes de un procedimiento CL

Mientras cada sentencia fuente entrada como parte de un procedimiento CL es en realidad un mandato CL, el fuente puede dividirse en las siguientes partes básicas utilizadas en muchos de los procedimientos CL habituales.

Mandato PGM
PGM PARM(&A)

Mandato optativo PGM al inicio del procedimiento que identifica los parámetros recibidos.

Mandatos de declaración
(**DCL, DCLF**)

Declaración obligatoria de las variables del programa cuando éstas se utilizan. Los mandatos de declaración deben preceder a todos los demás mandatos a excepción del mandato PGM.

Mandatos de proceso CL
CHGVAR, SNDPGMMSG, OVRDBF, DLTF, ...

Mandatos CL utilizados como sentencias fuente para manipular constantes o variables (lista parcial).

Mandatos de control lógico
IF, THEN, ELSE, DO, ENDDO, GOTO

Mandatos utilizados para controlar el proceso dentro del procedimiento CL.

Funciones incorporadas
%SUBSTRING (%SST), %SWITCH, y %BINARY (%BIN)

Funciones incorporadas y operadores utilizados en expresiones aritméticas, relacionales o lógicas.

Mandatos de control de programa
CALL, RETURN

Mandatos CL utilizados para pasar el control a otros programas.

Mandatos de control de procedimiento
CALLPRC, RETURN

Mandatos CL utilizados para pasar el control a otros programas.

Mandato ENDPGM
ENDPGM

Mandato opcional para Finalizar Programa.

La secuencia, combinación y extensión de estos componentes están determinadas por la lógica y el diseño de la aplicación.

Un procedimiento CL puede hacer referencia a otros objetos que deben existir cuando se crea el procedimiento, cuando se procesa el mandato, o ambos. Esta distinción se trata en el apartado "Acceso a Objetos en Programas y Procedimientos CL" en el tema 5.1 y en las secciones que tratan de objetos diversos. En algunos casos, para que el procedimiento se ejecute satisfactoriamente, puede que necesite:

- Un archivo de pantalla. Los archivos de pantalla se utilizan para dar formato a una información en una pantalla de dispositivo. Si su procedimiento utiliza una pantalla, el archivo de pantalla y el formato de registro deben entrarse y crearse utilizando el mandato Crear Archivo de Pantalla (CRTDSPF) antes de crear el procedimiento. También debe declararse al procedimiento de la sección DCL utilizando el mandato Declarar Archivo (DCLF). (Consulte el manual *Gestión de datos* para ver los mensajes e identificadores).

- Un archivo de base de datos. Un procedimiento CL puede leer los registros de un archivo de base de datos. Si su procedimiento utiliza un archivo de base de datos, el archivo debe crearse utilizando el mandato Crear Archivo Físico (CRTPF) o Crear Archivo Lógico (CRTLF) antes de crear el módulo. Puede utilizar las Especificaciones de Descripción de Datos (DDS), el Lenguaje de Consulta Estructurada (SQL) o el Programa de Utilidad de Definición Interactiva de Datos (IDDU) para definir el formato de los registros en el archivo. El archivo debe también declararse al procedimiento en la sección DCL utilizando el mandato Declarar Archivo (DCLF). Consulte la sección "Trabajar con Archivos en Procedimientos CL" en el tema 5.2 y el manual *DB2/400 Programación de la base de datos* para obtener más información.

- Otros programas. Si utiliza el mandato CALL, el programa al que se llama debe existir antes de que se ejecute dicho mandato. No tiene porque existir cuando el procedimiento de llamada se compila. Véase el apartado "Acceso a Objetos en Programas y Procedimientos CL" en el tema 5.1 y el Capítulo 3 para obtener más información.

- Otros procedimientos. Si utiliza el mandato CALLPRC, el procedimiento al que se llama debe existir cuando se ejecuta CRTPGM. No tiene porque existir cuando se ejecuta CRTCLMOD.

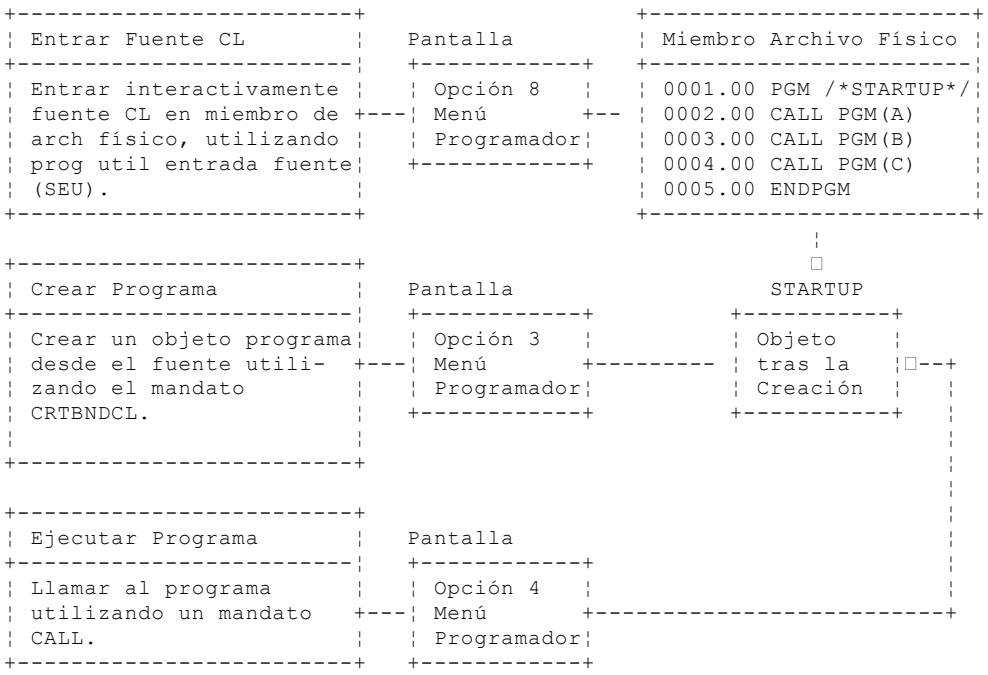
2.1.4 Ejemplo de un programa CL sencillo

Un programa CL puede ser tan sencillo o tan complejo como desee. Para consolidar diversas actividades realizadas normalmente por el operador de sistema al empezar la jornada (llamar a los programas A, B, y C, por ejemplo), puede crear un procedimiento CL STARTUP con el siguiente código:

```
  PGM  /*  STARTUP  */
  CALL PGM(A)
  CALL PGM(B)
  CALL PGM(C)
  ENDPGM
```

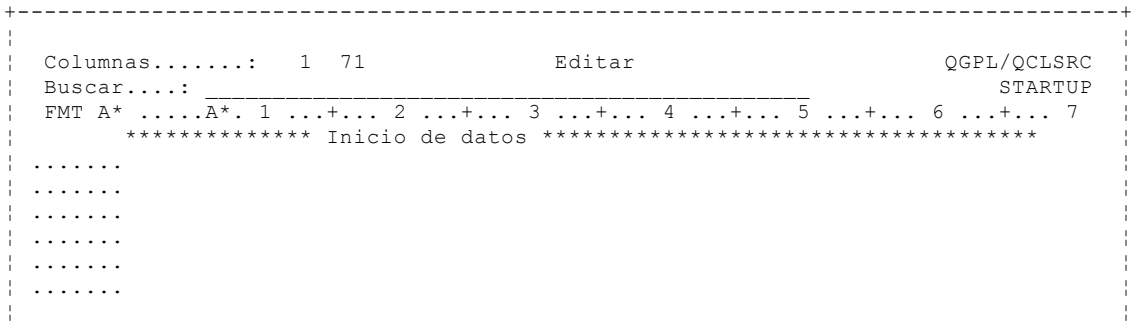
En este ejemplo, el Menú del Programador se utiliza para crear el programa. También puede utilizar el gestor para el desarrollo de programas (PDM), que forma parte del programa bajo licencia Juego de Herramientas para el Desarrollo de Aplicaciones/400.

Para entrar, crear y utilizar este programa, siga los pasos siguientes:



Para entrar el fuente CL:

- Seleccione la opción 8 (Editar fuente) en el Menú del Programador y especifique STARTUP en el campo Parm. Esta opción crea un miembro fuente denominado STARTUP que será también el nombre del programa.
- Especifique **CLLE** en el campo Tipo y pulse la tecla Intro.
- En la pantalla del SEU, utilice el mandato de línea **I** (insertar) para entrar los mandatos CL (CALL es un mandato CL).



Cuando acabe de entrar las sentencias fuente:

- Pulse F3 para salir de SEU.
- Acepte el valor por omisión que aparezca en la pantalla de salida (opción 2, Salir y actualizar miembro) y pulse la tecla Intro para

volver al Menú del Programador.

- Seleccione la opción 3 (Crear objeto) para crear un programa a partir de las sentencias fuente entradas. No tiene que cambiar ninguna otra información en la pantalla.

Nota: Los programas (A, B y C) a los que se hace referencia no tienen que existir cuando se crea el programa STARTUP.

Cuando el programa se ha creado, puede llamarse desde el Menú del Programador seleccionando la opción 4 (Llamar programa) y especificando STARTUP en el campo Parm. Sin embargo, si intenta ejecutar este programa de ejemplo, los programas referidos ya deben existir cuando se ejecuten los mandatos CALL.

2.2 Mandatos Utilizados en Procedimientos CL

Un procedimiento puede contener sólo mandatos CL. Éstos pueden ser suministrados por IBM o puede haberlos definido usted. Algunos mandatos suministrados por IBM *no* pueden utilizarse en procedimientos CL. Consulte la publicación CL Reference para ver las descripciones individuales de mandatos y su aplicabilidad en procedimientos CL.

Subtemas

2.2.1 Mandatos entrados en los parámetros RQSDTA y CMD

Algunos mandatos CL, como Transferir Trabajo (TFRJOB) y Someter Trabajo (SBMJOB) tienen parámetros RQSDTA o CMD que pueden utilizar otro mandato CL como parámetro. Los mandatos que se pueden utilizar solamente como sentencias fuente en procedimientos CL no pueden utilizarse como valores en el parámetro RQSDTA o CMD.

2.2.2 Mandatos CL

A continuación se facilita una lista de los mandatos utilizados con frecuencia en Procedimientos CL. Puede utilizarla para seleccionar el mandato adecuado para la función que desea realizar, así como para determinar qué mandato tendría que consultar en el manual CL Reference. Si se familiariza con la función de estos mandatos, le resultará más fácil comprender los restantes temas de este capítulo. El superíndice 1 indica los mandatos que pueden utilizarse *sólo* en procedimientos CL.

Función del sistema	**Mandato**	**Función del mandato**
Cambiar Control Procedimiento	CALL (Llamar)	Llama a un programa
	CALLPRC (Llamar Procedimiento)	Llama a un procedimiento
	RETURN (Regresar)	Regresa al mandato que está a continuación del mandato que originó la ejecución de un programa o procedimiento.
Límites procedimiento CL	PGM (Programa) (1)	Indica el principio del fuente de procedimiento CL
	ENDPGM (Fin Programa) (1)	Indica el final del fuente de procedimiento CL
Lógica Procedimiento CL	IF (If) (1)	Procesa mandatos en función del valor de una expresión lógica
	ELSE (Sino) (1)	Define la acción a tomar si no se cumple la condición de un mandato IF
	DO (Hacer) (1)	Indica el inicio de un grupo de acciones a realizar
	ENDDO (Finalizar Hacer) (1)	Indica el final de un grupo de acciones a realizar
	GOTO (Ir a) (1)	Bifurca hacia otro mandato
Variables Procedimiento	CHGVAR (Cambiar Variable)(1)	Cambia el valor de una variable CL
	DCL (Declarar) (1)	Declaración de una variable
Conversión	CHGVAR (Cambiar Variable)(1)	Cambia el valor de una variable CL
	CVTDAT (Convertir Fecha) (1)	Cambia el formato de una fecha
Áreas de datos	CHGDTAARA (Cambiar Área de Datos)	Cambia un área de datos
	CRTDTAARA (Crear Área de Datos)	Crea un área de datos
	DLTDTAARA (Suprimir Área de Datos)	Suprime un área de datos
	DSPDTAARA (Visualizar Área de Datos)	Visualiza un área de datos
	RTVDTAARA (Recuperar Área de Datos)	Copia el contenido de un área de datos en una variable CL
Archivos	ENDRCV (Finalizar Recep.) (1)	Cancela una petición de entrada emitida anteriormente por un mandato RCVF, SNDF o SNDRCVF a un archivo de pantalla

	DCLF (Declarar Archivo) (1)	Declara un archivo de pantalla o de base de datos
	RCVF (Recibir Archivo) (1)	Lee un registro de un archivo de pantalla o de base de datos
	RTVMBRD (Recuperar Descripción de Miembro) (1)	Recupera una descripción de un miembro determinado de un archivo de base de datos
	SNDF (Enviar Archivo) (1)	Graba un registro en un archivo de pantalla
	SNDRCVF (Enviar/Recibir Archivo) (1)	Graba un registro en un archivo de pantalla y lo lee después de que el usuario responda
	WAIT (Esperar) (1)	Espera a recibir los datos de un mandato SNDF, RCVF o SNDRCVF emitido para un archivo de pantalla
Mensajes	MONMSG (Supervisar Mensaje) (1)	Supervisa los mensajes de escape, estado y notificación enviados a una cola de mensajes del programa
	RCVMSG (Recibir Mensaje) (1)	Copia un mensaje de una cola de mensajes en las variables CL de un procedimiento CL
	RMVMSG (Eliminar Mensaje) (1)	Elimina un mensaje especificado de una cola de mensajes determinada
	RTVMSG (Recup. Mensaje) (1)	Copia un mensaje predefinido de un archivo de mensajes en variables de procedimiento CL
	SNDPGMMSG (Enviar Mensaje de Programa) (1)	Envía un mensaje de programa a una cola de mensajes
	SNDRPY (Enviar Respuesta) (1)	Envía un mensaje de respuesta al emisor de un mensaje de consulta
	SNDUSRMSG (Enviar Mensaje de Usuario)	Envía un mensaje informativo o de consulta a una estación de pantalla o al operador del sistema
Mandatos varios	CHKOBJ (Comprobar Objeto)	Comprueba la existencia de un objeto y, opcionalmente, la autorización necesaria para utilizar el objeto
	PRTCMDUSG (Imprimir Utilización del Mandato)	Genera una lista de referencias cruzadas para un grupo especificado de mandatos utilizados en un grupo determinado de procedimientos CL
	RTVCFGSRC (Recuperar Fuente de Configuración)	Genera el fuente de mandatos CL para crear objetos existentes de configuración y coloca el fuente en un miembro del archivo fuente
	RTVCFGSTS (Recuperar Estado de Configuración) (1)	Ofrece a las aplicaciones la posibilidad de recuperar el estado de configuración de tres objetos de configuración: línea, controlador y dispositivo
	RTVJOBA (Recuperar Atributos de Trabajo) (1)	Recupera el valor de uno o varios atributos de trabajo y los coloca en una variable CL
	RTVSYSVAL (Recuperar Valor del Sistema) (1)	Recupera un valor del sistema y lo coloca en una variable CL
	RTVUSRPRF (Recuperar Perfil de Usuario) (1)	Recupera los atributos de perfil de usuario y los coloca en las variables CL

Mandatos Creación Programa	CRTCLMOD (Crear Módulo CL)	Crea un módulo CL
	DLTMOD (Suprimir Módulo)	Suprime un módulo
	DLTPGM (Suprimir Programa)	Suprime un programa
	CRTBNDCL (Crear Programa Lenguaje de Control Enlazado)	Crea un programa CL enlazado
	CRTPGM (Crear Programa)	Crea un programa
	CRTSRVPGM (Crear Programa Servicio)	Crea un programa de servicio.

2.3 Utilización de Procedimientos CL

La programación CL es un instrumento flexible que le permite realizar una gran variedad de operaciones. Cada uno de los usos citados a continuación se describe con mayor detalle en los apartados posteriores de este capítulo. En general, puede:

- Utilizar variables, mandatos de control lógico, expresiones y funciones incorporadas para manipular y procesar datos en un procedimiento CL:

```
PGM
DCL &C *LGL
DCL &A *DEC VALUE(22)
DCL &B *CHAR VALUE(ABCDE)
□
□
□
CHGVAR &A (&A + 30)
□
□
□
IF (&A < 50) THEN(CHGVAR &C '1')
□
DSPLIB ('Q' || &B)
□
IF (%SST(&B 5 1)=E) THEN(CHGVAR &A 12)
□
□
□
ENDPGM
```

- Utilizar un valor del sistema como variable en un procedimiento CL.

```
                            PGM
                            DCL &TIME *CHAR 6
                            □
Valores del sistema         □
+---------+                 □
¦  QTIME  +---------------  RTVSYSVAL QTIME &TIME
¦  QDATE  ¦                 □
¦    .    ¦                 □
¦    .    ¦                 □
¦    .    ¦                 ENDPGM
+---------+
```

- Utilizar un atributo de trabajo como variable en un procedimiento CL.

```
                            PGM
                            DCL &USER *CHAR 10
Atributos Trabajo           □
+------------+              □
¦ Nomb Trab  ¦              □
¦ Nomb Usuar +------------  RTVJOBA USER(&USER)
¦ Núm Trabaj ¦              □
¦     .      ¦              □
¦     .      ¦              □
¦     .      ¦              ENDPGM
+------------+
```

- Enviar y recibir datos a y desde un archivo de pantalla con un procedimiento CL.

```
+------------------+
¦ Especificaciones ¦
¦ de Descripción   ¦
¦ de Datos         ¦
¦      (DDS)       ¦
+------------------+
         ¦
         □                  PGM
+------------------+        DCL FILE(DISPLAY)
¦    Pantalla      ¦        DCL &OPTION *CHAR 2
¦                  ¦        □
¦                  ¦        □
¦                  ¦        □
¦                  ¦        RCVF ...
¦ Opción __  ------+-----   IF (&OPTION *EQ 1) THEN(CALL PGMA)
+------------------+        □
                            □
                            □
                            ENDPGM
```

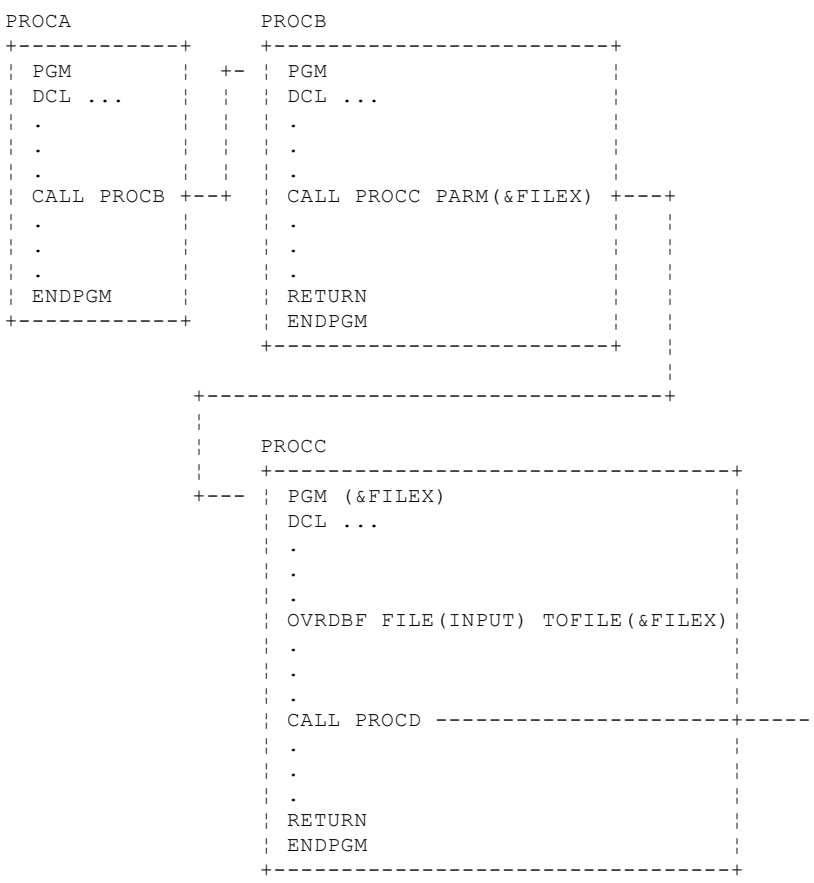
- Crear un procedimiento CL para supervisar mensajes de error para un

trabajo, y realizar las acciones correctivas si es necesario.

```
            PGM

            MONMSG MSGID(CPF0001) EXEC(GOTO ERROR)
            CALL PROGA
            CALL PROGB
            RETURN
      ERROR: SNDPGMMSG MSG('Mandato CALL falló') MSGTYPE(*ESCAPE)
            ENDPGM
```

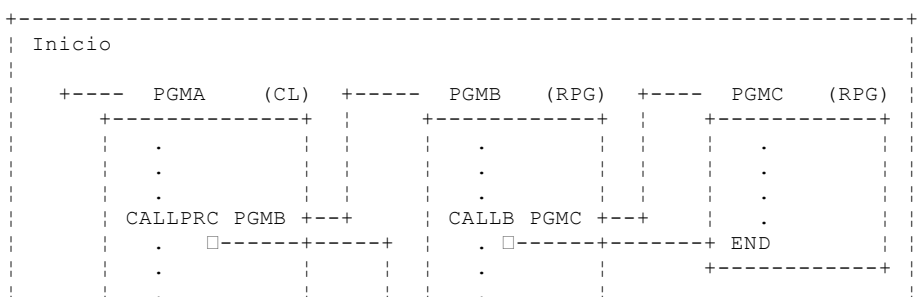
- Controlar el proceso entre los procedimientos y programas, pasar parámetros de un procedimiento CL a otros procedimientos o programas para alterar temporalmente los archivos.

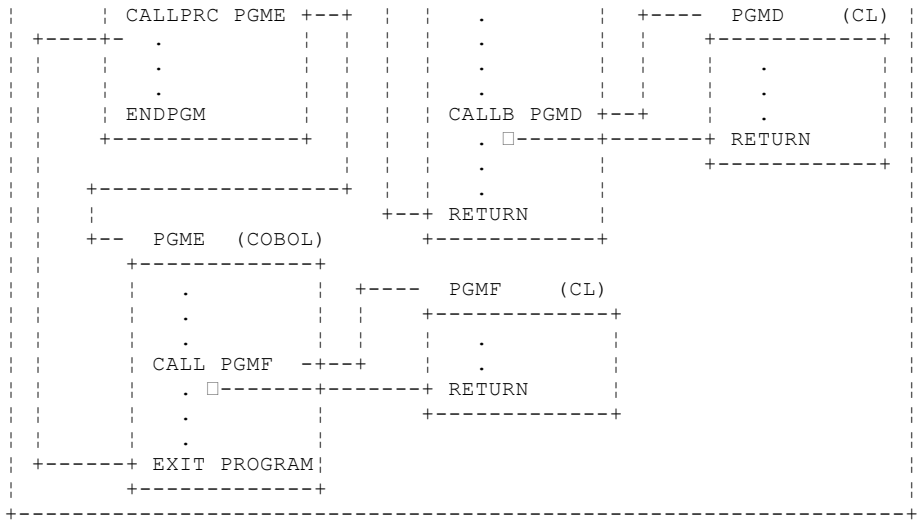


Utilización de Procedimientos CL para Controlar el Proceso

Utilizado como procedimiento de control, un procedimiento CL puede llamar a procedimientos escritos en otros lenguajes. La siguiente ilustración muestra cómo se puede pasar el control entre un procedimiento CL y procedimientos RPG* e ILE COBOL/400* en una aplicación. Para utilizar la aplicación, el usuario de una estación de trabajo solicitaría el programa A, que controla toda la aplicación. La ilustración muestra: El ejemplo anterior muestra un solo programa enlazado (PGMA) que se llama utilizando el mandato CALL con PGMA. PGMA consta de:

- Un procedimiento CL (PGMA) que llama a un procedimiento RPG IV (PGMB)
- Un procedimiento RPG (PGMB) que llama a otro procedimiento RPG IV (PGMC)
- Un procedimiento RPG IV (PGMB) que llama a un procedimiento CL (PGMD)
- Un procedimiento CL (PGMA) que llama a un procedimiento ILE COBOL/400 (PGME)
- Un programa ILE COBOL/400 (PGME) que llama a un procedimiento CL (PGMF)





Los procedimientos pueden crearse tal como se indica en el siguiente ejemplo. Puede entrar un fuente para procedimientos en miembros fuente individuales.

```
      CRTCLMOD PGMA
      CRTRPGMOD PGMB
      CRTRPGMOD PGMC
      CRTCLMOD PGMD
      CRTCBLMOD PGME
      CRTCLMOD PGMF
      CRTPGM PGM(PGMA) +
             MODULE(PGMA PGMB PGMC PGMD PGME PGMF) +
             ENTMOD(*FIRST)
```

2.4 Trabajar con Variables

Los procedimientos CL constan de mandatos CL, y los mandatos en sí están formados por la sentencia del mandato, parámetros y valores de parámetros. Las reglas de sintaxis para escribir mandatos se explican en el manual CL Reference.

Los valores de los parámetros pueden expresarse como variables, constantes o expresiones. Una **variable** es un valor que tiene un nombre y que se puede cambiar, y al que se puede acceder o que se puede modificar haciendo referencia a dicho nombre. Las variables pueden utilizarse como sustitutos de la mayoría de valores de parámetros en los mandatos CL. Cuando se especifica una variable CL como valor de parámetro y se ejecuta el mandato que lo contiene, el valor de la variable se utiliza como el valor del parámetro. Cada vez que se ejecuta el mandato, puede modificarse el valor de la variable. Las variables y las expresiones sólo se pueden utilizar como valores de parámetros en programas y procedimientos CL.

Las variables no se almacenan en bibliotecas; no son objetos; y sus valores se destruyen cuando ya no se llama al procedimiento que las contiene. Los nombres de variable en procedimientos CL deben empezar por el símbolo & (ampersand) seguido de como máximo 10 caracteres. El primer carácter después del símbolo & ha de ser alfabético y el resto de caracteres alfanuméricos, por ejemplo, &FILE o &DSPFL1. El uso de variables como valores da a la programación CL una flexibilidad especial, ya que esto permite la manipulación a alto nivel de objetos, cuyo contenido puede verse modificado por aplicaciones específicas. Puede, por ejemplo, escribir un procedimiento CL para dirigir el proceso de otros programas o la operación de varias estaciones de trabajo sin especificar qué programas o estaciones de trabajo van a controlarse. Éstos estarían identificados en el procedimiento CL como variables. El valor de las variables se puede definir (especificarse) cuando se ejecuta el procedimiento CL.

Todas las variables deben estar declaradas (definidas) en el procedimiento CL antes de que éste las pueda utilizar:

- Declarar variable. La definición se lleva a cabo utilizando el mandato Declarar Variable CL (DCL), y consiste en definir los atributos de la variable, que son tipo, longitud y valor inicial.

```
DCL  VAR(&AREA)  TYPE(*CHAR)  LEN(4)  VALUE(BOOK)
```

- Declarar archivo. Si su procedimiento CL utiliza un archivo, debe especificar el nombre del archivo en el parámetro FILE del mandato Declarar Archivo (DCLF). El archivo contiene una descripción (formato) de los registros del archivo y los campos de los registros. Durante la compilación, el mandato DCLF declara implícitamente variables CL para los campos e indicadores definidos en el archivo.

  Si las DDS para el archivo tienen un registro con dos campos (F1 y F2), entonces se declaran automáticamente dos variables, &F1 y &F2, en el programa.

```
DCLF  FILE(MCGANN/GUIDE)
```

  Si el archivo es un archivo físico que se creó sin DDS, se declara una variable para todo el registro. La variable tiene el mismo nombre que el archivo y su longitud es la misma que la longitud de registro del archivo.

Los mandatos de declaración deben preceder a los demás mandatos del procedimiento (excepto el mandato PGM), pero pueden combinarse en cualquier orden.

Además de las utilizaciones que se han tratado en este apartado, las variables pueden utilizarse para:

- Pasar información entre procedimientos y trabajos. Consulte el Capítulo 3, "Control del Flujo y Comunicación entre Programas y Procedimientos" en el tema 3.0.

- Pasar información entre procedimientos y pantallas. Consulte el "Trabajo con Archivos de Pantalla de Múltiples Dispositivos" en el tema 5.2.7.

- Procesar mandatos condicionalmente. Véase el apartado "Control del Proceso en un Procedimiento CL" en el tema 2.5.

- Crear objetos. Una variable puede utilizarse en lugar de un nombre de objeto o nombre de biblioteca o en ambos casos. El ejemplo siguiente

muestra el mandato Crear Archivo Físico (CRTPF) utilizado con una biblioteca especificada en la primera línea y con una variable que sustituye el nombre de la biblioteca en la segunda línea:

```
CRTPF  FILE(DSTPRODLB/&FILE)
CRTPF  FILE(&LIB/&FILE)
```

No pueden utilizarse variables para cambiar un nombre o palabra clave de mandato o para especificar un nombre de procedimiento para el mandato CALLPRC. Sin embargo, los parámetros de los mandatos se pueden cambiar durante el proceso de un procedimiento CL utilizando la función de solicitud. Véase el apartado "Permitir al Usuario Cambiar Mandatos CL en la Ejecución" en el tema 6.5 para obtener más información.

También es posible agrupar las palabras clave y los parámetros de un mandato y procesarlo utilizando la función QCAPCMD. Consulte la sección "Utilización del Programa QCAPCMD" en el tema 6.1 para obtener más información sobre QCAPCMD.

Subtemas

2.4.1 Declaración de una Variable

En el formato más sencillo, el mandato Declarar Variable CL (DCL) tiene los siguientes parámetros:

IMAGEN 1

Al utilizar un mandato DCL, debe seguir las normas siguientes:

- El nombre de la variable CL debe empezar con un símbolo (&) seguido por un máximo de 10 caracteres. El primer carácter a continuación de & debe ser alfabético, y el resto de los caracteres alfanuméricos. Por ejemplo, &PART.

- El valor de la variable CL debe ser uno de los siguientes:

  - Una serie de caracteres de 5000 posiciones de longitud como máximo.
  - Un valor decimal empaquetado que totaliza 15 dígitos con 9 posiciones decimales como máximo.
  - Un valor lógico de '0' ó '1', donde '0' puede significar desactivado, falso o no, y '1' puede significar activado, verdadero o sí. Una variable lógica tiene que ser '0' ó '1'.

- Si no se especifica ningún valor inicial, entonces se supone lo siguiente:

  - 0 para variables decimales
  - Espacios en blanco para variables de tipo carácter
  - '0' para variables lógicas

  Para tipos decimales y de caracteres, si especifica un valor inicial y no especifica el parámetro LEN, la longitud por omisión es la misma que la longitud del valor inicial. Para el tipo *CHAR, si no especifica el parámetro LEN, la serie puede ser de 5000.

- Declare los parámetros como variables en las sentencias DCL del programa.

2.4.2 Utilización de Variables para Especificar una Lista o Nombre Calificado

El valor de un parámetro puede ser una lista. Por ejemplo, el mandato Cambiar Lista de Bibliotecas (CHGLIBL) requiere una lista de bibliotecas en el parámetro LIBL, separadas por espacios en blanco. Los elementos de esta lista pueden ser variables:

```
  CHGLIBL LIBL(&LIB1 &LIB2 &LIB3)
```

Cuando las variables se utilizan para especificar elementos en una lista, cada elemento debe declararse por separado:

```
  DCL VAR(&LIB1) TYPE(*CHAR) LEN(10) VALUE(QTEMP)
  DCL VAR(&LIB2) TYPE(*CHAR) LEN(10) VALUE(QGPL)
  DCL VAR(&LIB3) TYPE(*CHAR) LEN(10) VALUE(DISTLIB)
  CHGLIBL LIBL(&LIB1 &LIB2 &LIB3)
```

Los elementos de la variable no pueden especificarse en una lista como una serie de caracteres:

Incorrecto:

```
  DCL   VAR(&LIBS) TYPE(*CHAR) LEN(20) +
        VALUE('QTEMP QGPL DISTLIB')
  CHGLIBL LIBL(&LIBS)
```

Cuando se presenta como un sola serie de caracteres, el sistema no visualiza la lista como si constase de elementos separados y puede producirse un error.

También puede utilizar variables para especificar un nombre calificado si cada calificador se declara como una variable distinta:

```
  DCL VAR(&PGM) TYPE(*CHAR) LEN(10)
  DCL VAR(&LIB) TYPE(*CHAR) LEN(10)
  CHGVAR VAR(&PGM) VALUE(MYPGM)
  CHGVAR VAR(&LIB) VALUE(MYLIB)
  .
  .
  .
  DLTPGM PGM(&LIB/&PGM)
  ENDPGM
```

En este ejemplo, el nombre del programa y de la biblioteca se declaran por separado. Estos nombres no se pueden especificar en una sola variable, como en el ejemplo siguiente:

Incorrecto:

```
  DCL VAR(&PGM) TYPE(*CHAR) LEN(10)
  CHGVAR VAR(&PGM) VALUE('MYLIB/MYPGM')
  DLTPGM PGM(&PGM)
```

En este caso, el sistema también considera el valor como una sola serie de caracteres, no como dos objetos (una biblioteca y un objeto). Si un nombre calificado ha de manejarse como una única variable con un valor de serie de caracteres, puede utilizar la función incorporada %SUBSTRING y la función de concatenación *TCAT para asignar nombres de objeto y de biblioteca a variables separadas. Véanse los apartados "Utilización de la Función Incorporada %SUBSTRING" en el tema 2.5.8 y Capítulo 9 para consultar ejemplos en los que se utiliza la función %SUBSTRING.

2.4.3 Caracteres en Minúscula en las Variables

Los valores reservados, como *LIBL, que pueden utilizarse como variables han de estar siempre en mayúsculas, especialmente si se presentan como series de caracteres entre apóstrofos. Por ejemplo, si desea sustituir una variable por un nombre de biblioteca en un mandato, el código correcto es el siguiente:

```
  DCL VAR(&LIB) TYPE(*CHAR) LEN(10) VALUE('*LIBL')
  DLTPGM &LIB/MYPROG
```

Sin embargo, sería *incorrecto* especificar el parámetro VALUE del modo siguiente:

```
  DCL VAR(&LIB) TYPE(*CHAR) LEN(10) VALUE('*libl')
```

Fíjese que si el parámetro VALUE no se hubiera escrito entre apóstrofos, sería correcto porque sin los apóstrofos se convertiría automáticamente a mayúsculas. Este error se produce con frecuencia cuando el parámetro se pasa como entrada a un programa o procedimiento desde una pantalla como una serie de caracteres y la entrada de pantalla se efectúa en minúsculas.

2.4.4 Variables que Sustituyen Valores de Parámetros Numéricos o Reservados

Algunos mandatos CL permiten valores numéricos o predefinidos (reservados) en ciertos parámetros. Cuando esto ocurre, también puede utilizar variables de tipo carácter para representar el valor del parámetro de un mandato.

Cada parámetro en un mandato puede aceptar solamente ciertos tipos de valores. El parámetro puede permitir como valores un entero, una serie de caracteres, un valor reservado, una variable de un tipo especificado o una combinación de todos ellos. Algunos parámetros requieren ciertos tipos de valores. Si el parámetro permite valores numéricos (si el valor está definido en el mandato como *INT2, *INT4 o *DEC) y permite también valores reservados (una serie de caracteres precedida por un asterisco), entonces puede utilizar una variable como valor para el parámetro. La variable debe declararse como TYPE(*CHAR) si tiene la intención de utilizar un valor reservado.

Por ejemplo, el mandato Cambiar Cola de Salida (CHGOUTQ) tiene un parámetro separador de trabajos (JOBSEP) que puede tener un valor numérico (de 0 a 9) o el valor por omisión predefinido, *SAME. Dado que ambos, el número y el valor predefinido son aceptables, puede escribir también un procedimiento CL que sustituya una variable de tipo carácter para el valor JOBSEP:

```
        PGM
        DCL &NRESP *CHAR LEN(6)
        DCL &SEP *CHAR LEN(4)
        DCL &FILNAM *CHAR LEN(10)
        DCL &FILLIB *CHAR LEN(10)
        DCLF.....
        .
        .
        .
  LOOP: SNDRCVF.....
        IF (&SEP *EQ IGNR) GOTO END
        ELSE IF (&SEP *EQ NONE) CHGVAR &NRESP '0'
        ELSE IF (&SEP *EQ NORM) CHGVAR &NRESP '1'
        ELSE IF (&SEP *EQ SAME) CHGVAR &NRESP '*SAME'
        CHGOUTQ OUTQ(&FILLIB/&FILNAM) JOBSEP(&NRESP)
        GOTO LOOP
  END:  RETURN
        ENDPGM
```

En el ejemplo anterior, el usuario de la estación de pantalla entra información que describe el número de separadores de trabajo que desea utilizar en una cola de salida especificada. La variable &NRESP es una variable de tipo carácter que manipula valores numéricos y predefinidos (observe el uso de apóstrofos). El parámetro JOBSEP del mandato CHGOUTQ reconocerá estos valores como si se hubieran entrado como valores numéricos o predefinidos. Las DDS del archivo de pantalla utilizado en este programa debe utilizar la palabra clave VALUES para restringir las respuestas del usuario a IGNR, NONE, NORM o SAME.

Si el parámetro permite valores numéricos (*INT2, *INT4 o *DEC) y no desea entrar valores reservados (como *SAME), puede utilizar una variable decimal en este parámetro.

Para obtener más información acerca de tipos de valores permitidos por los parámetros de mandato, consulte el Capítulo 9 de este manual y la publicación CL Reference.

Otra alternativa de esta función es utilizar el programa de solicitud en los procedimientos CL.

2.4.5 Cambio del Valor de una Variable

Puede cambiar el valor de una variable CL utilizando el mandato Cambiar Variable (CHGVAR). Este valor se puede cambiar:

- Por una constante:

```
  CHGVAR   VAR(&INVCMPLT)   VALUE(0)
```

  o

```
  CHGVAR   &INVCMPLT  0
```

  &INVCMPLT se establece en 0.

- Por el valor de otra variable:

```
  CHGVAR   VAR(&A)   VALUE(&B)
```

  o

```
  CHGVAR   &A  &B
```

  En &A se establece el valor de la variable &B.

- Por el valor de una expresión después de evaluarla:

```
  CHGVAR   VAR(&A)   VALUE(&A + 1)
```

  o

```
  CHGVAR   &A  (&A + 1)
```

  El valor de &A se incrementa en 1.

- Por el valor producido por la función incorporada %SST (véase el apartado "Utilización de la Función Incorporada %SUBSTRING" en el tema 2.5.8 para más información):

```
  CHGVAR   VAR(&A)   VALUE(%SST(&B 1 5))
```

  En &A se establecen los primeros 5 caracteres del valor de la variable &B.

- Por el valor producido por la función incorporada %SWITCH (véase el apartado "Utilización de la Función Incorporada %SWITCH" en el tema 2.5.9 para más información):

```
  CHGVAR  VAR(&A)  VALUE(%SWITCH(0XX111X0))
```

  En &A se establece 1 si los conmutadores de trabajo 1 y 8 valen 0, y los conmutadores 4, 5 y 6 valen 1; de lo contrario &A se establece en 0.

- Por el valor producido por la función incorporada %BIN (véase el apartado "Utilización de la Función Incorporada %BINARY" en el tema 2.5.7 para más información):

```
  CHGVAR  VAR(&A)  VALUE(%BIN((%B 1 4))
```

  Los cuatro primeros caracteres de la variable &B se convierten al equivalente decimal y se almacenan en la variable decimal &A.

El mandato CHGVAR también puede utilizarse para recuperar y para cambiar el área de datos local. Por ejemplo, los mandatos siguientes borran 10 bytes del área de datos local y recuperan parte de dicha área:

```
CHGVAR %SST(*LDA 1 10) ' '

CHGVAR &A %SST(*LDA 1 10)
```

En el caso de una variable lógica, el valor que va a tomar la variable debe ser un valor lógico. Para variables decimales, puede utilizarse un valor decimal o de tipo carácter. En las variables de tipo carácter se aceptan valores decimales o de tipo carácter.

Al especificar un valor decimal para una variable de tipo carácter, recuerde lo siguiente:

- El valor de la variable de tipo carácter se justifica por la derecha y, en caso necesario, se rellena con ceros iniciales.
- La variable de tipo carácter debe tener una longitud suficiente para contener una coma decimal y un signo menos (-), cuando sea necesario.
- Cuando se utilice, se coloca un signo menos (-) en la posición que esté más a la izquierda del valor.

Por ejemplo &A es una variable tipo carácter a la que se cambia el valor por el de la variable de tipo decimal &B. La longitud de &A es 6. La longitud y las posiciones decimales de &B son 5 y 2, respectivamente. El valor actual de &B es **123**. El valor resultante de &A es **123.00**.

Cuando se especifica un valor de caracteres para una variable decimal, recuerde lo siguiente:

- La coma decimal está determinada por la colocación de una coma decimal en el valor de caracteres. Si este valor no contiene una coma decimal, ésta se coloca en la posición que hay más a la derecha del valor.

- El valor de caracteres puede contener un signo menos (-) o más (+) inmediatamente a la izquierda del valor; no se permiten espacios en blanco entre ellos. Si el valor de caracteres no tiene signo, se supone que el valor es positivo.

- Si el valor de caracteres contiene más caracteres a la derecha de la coma decimal que los que pueden haber en la variable decimal, los caracteres se truncan. Sin embargo, si el exceso de caracteres se da a la izquierda de la coma decimal, no se truncan y se produce un error.

  Por ejemplo, &C es una variable tipo carácter a la que se cambia el valor por el de la variable tipo carácter &D. La longitud de &C es 5, con dos posiciones decimales. La longitud de &D es 10 y su valor actual es +123.1bbbb (donde b=blanco). El valor resultante de &C es **123.10**.

2.4.6 Blancos de Cola en Parámetros de Mandato

Algunos parámetros de mandatos se definen con el parámetro VARY(*YES). Este valor de parámetro hace que la longitud del valor que se ha pasado sea el número de caracteres entre apóstrofos. Cuando una variable CL se utiliza para especificar el valor de un parámetro definido de esta forma, el sistema elimina los blancos de cola antes de determinar la longitud de la variable que se pasa al programa que procesa el mandato. Si los blancos de cola están presentes en el parámetro y son significativos para el mismo, debe tomar acciones especiales para asegurarse de que la longitud que se pasa los incluye. La mayoría de parámetros de mandatos se definen y utilizan de forma que pueda darse esta condición. Un ejemplo de un parámetro definido donde es probable de que se produzca esta condición es el elemento de valor clave del parámetro POSITION del mandato OVRDBF.

Cuando se dé esta condición, puede obtenerse el resultado deseado para estos parámetros creando una serie de mandatos que delimite el valor del parámetro con apóstrofos y pasando la serie a QCMDEXC o QCAPCMD para efectuar el proceso.

A continuación tiene un ejemplo de un programa que puede utilizarse para ejecutar el mandato OVRDBF en el cual los blancos de cola se van a incluir como parte del valor clave. Esta misma técnica puede utilizarse para otros mandatos que tienen parámetros definidos utilizando el parámetro VARY(*YES); los blancos de cola deben pasarse con el parámetro.

```
             PGM        PARM(&KEYVAL &LEN)
   /*   PROGRAMA PARA MOSTRAR CÓMO ESPECIFICAR VALOR CLAVE CON BLANCOS  */
   /*      DE COLA COMO PARTE DEL PARÁMETRO POSICIÓN EN EL MANDATO      */
   /*      OVRDBF EN UN PROGRAMA CL.                                    */
   /*   EL ELEMENTO VALOR CLAVE DEL PARÁMETRO POSICIÓN DEL MANDATO      */
   /*      OVRDBF SE DEFINE CON EL PARÁMETRO VARY(*YES).                */
   /*      LA DESCRIPCIÓN DE ESTE PARÁMETRO EN LA SENTENCIA DEFINICIÓN  */
   /*      DE MANDATO DE ELEMENTO INDICA QUE SI SE ESPECIFICA UN PARÁM. */
   /*      DEFINIDO DE ESTA FORMA COMO UNA VARIABLE CL LA LONGITUD SE   */
   /*      PASA COMO VARIABLE CON LOS BLANCOS DE COLA ELIMINADOS.       */
   /*      PUEDE UTILIZARSE UNA LLAMADA A QCMDEXC ENTRE APÓSTROFOS PARA */
   /*      DELIMITAR LA LONGITUD DEL VALOR CLAVE PARA ELUDIR ESTA       */
   /*      ACCIÓN                                                       */
   /*    PARÁMETROS--                                                   */
            DCL        VAR(&KEYVAL) TYPE(*CHAR) LEN(32)  /*  EL VALOR  +
                         DE CLAVE SOLICITADA. SE DEFINE COMO DE      +
                         32 CAR.  */
            DCL        VAR(&LEN) TYPE(*DEC) LEN(15 5)  /* LA LONGITUD +
                         DEL VALOR CLAVE A UTILIZAR. PUEDE UTILIZAR +
                         CUALQUIER VALOR DE 1 A 32  */
   /*   LA SERIE A TERMINAR PARA QUE EL MANDATO DE ALTERACIÓN TEMPORAL  */
   /*        SE PASE A QCMDEXC (ANOTE 2 APÓSTROFOS PARA OBTENER UNO).   */
            DCL        VAR(&STRING) TYPE(*CHAR) LEN(100) +
                VALUE('OVRDBF FILE(X3) POSITION(*KEY 1 FMT1  '' ')
   /*   MARCADOR POSICIÓN 123456789 123456789 123456789 123456789      */
            DCL        VAR(&END) TYPE(*DEC) LEN(15 5)  /* UNA VARIABLE +
                         PARA CALCULAR EL FINAL DE LA CLAVE EN &STRING */

            CHGVAR     VAR(%SST(&STRING 40 &LEN)) VALUE(&KEYVAL)  /*  +
                         PONGA VALOR CLAVE EN LA SERIE DE MANDATO PARA +
                         QCMDEXC INMEDIATAMENTE DESPUÉS DEL APÓSTROFO */
            CHGVAR     VAR(&END) VALUE(&LEN + 40) /* POSICION DESPUES +
                         ÚLTIMO CARÁCTER DEL VALOR DE CLAVE  */
            CHGVAR     VAR(%SST(&STRING &END 2)) VALUE(''')')  /*  PONGA +
                         UN APÓSTROFE DE CIERRE Y PARÉNTESIS PARA +
                         FINALIZAR EL PARÁMETRO  */
            CALL       PGM(QCMDEXC) PARM(&STRING 100)  /*  LLAMAR  +
                         PARA PROCESAR EL MANDATO   */
            ENDPGM
```

Nota: Si utiliza VARY(*YES) y RTNVAL(*YES) y está pasando una variable CL, se pasa la longitud de la variable en lugar de la longitud de los datos en la variable CL.

2.4.7 Escribir Comentarios en Procedimintos CL

Cuando desee escribir comentarios en sus procedimientos CL o añadir comentarios a los mandatos de sus procedimientos, utilice las parejas de caracteres **/*** y ***/**. El comentario se ha de escribir entre estos símbolos.

El delimitador inicial de comentario, **/*** , requiere tres caracteres a menos que los caracteres **/*** aparezcan en las dos primeras posiciones de la serie del mandato. En este último caso, **/*** puede utilizarse sin ningún blanco posterior antes de un mandato.

Se pueden introducir los delimitadores iniciales del comentario de tres caracteres de una de las formas siguientes (b representa un espacio en blanco):

```
/*b
b/*
/**
```

Por lo tanto, el delimitador de comentario inicial puede entrarse de cuatro formas. El delimitador inicial de comentario, **/***, puede:

- Empezar en la primera posición de la serie de mandatos
- Estar precedido por un espacio en blanco
- Estar seguido por un espacio en blanco
- Estar seguido por un asterisco (**/****).

Por ejemplo, en el procedimiento siguiente, se han escrito comentarios para describir las posibles respuestas de usuario a un conjunto de opciones de menú:

```
           PGM /* ORD040C Menú dept general pedidos */
           DCLF FILE(ORD040CD)
  START:    SNDRCVF RCDFMT(MENU)
           IF (&RESP=1) THEN(CALL CUS210)
           /*Consulta cliente */
           ELSE +
                 IF (&RESP=2) THEN(CALL ITM210)
                 /**Consulta artículo*/
                 ELSE +
                       IF (&RESP=3) THEN(CALL CUS210)
                        /* Búsqueda nombre cliente */
                       ELSE +
                             IF (&RESP=4) THEN(CALL ORD215)
                             /** Pedidos por cliente */
                             ELSE +
                                   IF (&RESP=5) THEN(CALL ORD220)
                                   /* Pedido existente */
                                   ELSE +
                                         IF (&RESP=6) THEN(CALL ORD410C)
                                         /** Entrada pedidos */
                                         ELSE +
                                               IF (&RESP=7) THEN(RETURN)
           GOTO START
           ENDPGM
```

2.5 Control del Proceso en un Procedimiento CL

Los mandatos de un procedimiento CL se procesan con orden consecutivo. Cada mandato se procesa, uno tras otro, en la secuencia en la que se encuentran. Puede alterar este proceso consecutivo utilizando mandatos que cambian el flujo de la lógica en el procedimiento. Estos mandatos pueden ser condicionales (IF) o incondicionales (GOTO).

La bifurcación incondicional significa que pueden darse instrucciones al proceso para dirigirse a los mandatos o conjuntos de mandatos situados en cualquier parte del procedimiento, sin tener en cuenta las condiciones existentes en el momento en que se procesa la instrucción para bifurcar. Esta operación puede efectuarse con el mandato GOTO.

La bifurcación condicional significa que, bajo ciertas condiciones, el proceso puede bifurcarse a secciones o mandatos del procedimiento que no son consecutivos. Se puede bifurcar a cualquier sentencia del procedimiento. Esta acción se denomina proceso condicional porque la bifurcación sólo se produce cuando se cumple la condición especificada. El proceso condicional normalmente está asociado al mandato IF. Con el mandato ELSE, puede especificar el proceso alternativo a realizar si la condición no se cumple.

El mandato DO le permite crear grupos de mandatos que siempre se procesan juntos bajo condiciones especificadas.

Subtemas

2.5.1 Utilización del Mandato GOTO y las Etiquetas

El mandato GOTO procesa una bifurcación incondicional. Con este mandato, el proceso se dirige a otra parte del procedimiento (identificada mediante una etiqueta) siempre que se encuentra el mandato GOTO. Esta bifurcación no depende de la evaluación de una expresión. Tras la bifurcación a la sentencia que contiene la etiqueta, el proceso comienza en esta sentencia y continúa en una secuencia consecutiva; no vuelve al mandato GOTO a menos que lo especifique otra instrucción. La bifurcación puede dirigirse hacia adelante o hacia atrás. No puede utilizar GOTO para ir a una etiqueta fuera del procedimiento. El mandato GOTO tiene un parámetro, que contiene la etiqueta de la sentencia hacia la que se efectúa la bifurcación:

```
  GOTO CMDLBL(etiqueta)
```

Una etiqueta identifica la sentencia del procedimiento a la que el mandato GOTO dirige el proceso. Para utilizar un mandato GOTO, el mandato al que se bifurca debe tener una etiqueta.

```
            PGM
            .
            .
            .
  START:    SNDRCVF RCDFMT(MENU)
            IF (&RESP=1) THEN(CALL CUS210)
            .
            .
            .
            GOTO START
            .
            .
            .
            ENDPGM
```

La etiqueta utilizada en este ejemplo es START. Una etiqueta puede tener 10 caracteres como máximo y debe ir seguida por dos puntos sin espacios en blanco entre ellos, aunque sí puede haberlos entre la etiqueta y el nombre del mandato.

2.5.2 Utilización del Mandato IF

|El mandato IF se utiliza para declarar una condición que, si es verdadera, |especifica la ejecución de otra sentencia o grupo de sentencias del procedimiento a ejecutar. El mandato ELSE puede utilizarse con el mandato IF para especificar una sentencia o un grupo de sentencias a ejecutar si la condición expresada por el mandato IF es falsa.

El mandato incluye una expresión, que se comprueba (verdadera o falsa) y un parámetro THEN que especifica la acción a tomar si la expresión es cierta. El formato del mandato IF es el siguiente:

```
  IF COND(expresión-lógica) THEN(mandato-CL)
```

|La expresión lógica en el parámetro COND debe ser una variable lógica o |constante individual, o esta expresión debe describir una relación entre dos o mas operandos; entonces esta se evalúa como cierto o falso. Véase el apartado "Utilización de los Operadores *AND, *OR y *NOT" en el tema 2.5.6 para obtener información más detallada sobre la construcción de expresiones lógicas.

Si la condición descrita por la expresión lógica se evalúa como verdadera, el procedimiento procesa el mandato CL en el parámetro THEN. Este puede ser un sólo mandato o un grupo de mandatos (vea el apartado "Utilización del mandato DO y de grupos DO" en el tema 2.5.3). Si la condición no es verdadera, el procedimiento ejecuta el siguiente mandato secuencial.

Tanto COND como THEN son palabras clave en el mandato y pueden omitirse para la entrada posicional. A continuación presentamos ejemplos sintácticamente correctos de este mandato:

```
  IF COND(&RESP=1) THEN(CALL CUS210)
  IF (&A *EQ &B) THEN(GOTO LABEL)
  IF (&A=&B) GOTO LABEL
```

Se precisan espacios en blanco entre el nombre del mandato (IF) y la palabra clave (COND) o valor (&A...). No se permiten espacios en blanco entre la palabra clave, si se especifica, y el paréntesis izquierdo que encierra el valor.

A continuación se facilita un ejemplo de proceso condicional con un mandato IF. El proceso se bifurca de formas diferentes dependiendo de la evaluación de la expresión lógica en los mandatos IF. Suponga, por ejemplo, que en el inicio del código siguiente, el valor de &A es 2 y el valor de &C es 4.

```
         IF (&A=2) THEN(GOTO FINAL)
         IF (&A=3) THEN(CHGVAR &C 5)
         .
         .
         .
  FINAL: IF (&C=5) CALL PROGA
         ENDPGM
```

En este caso, el procedimiento procesa el primer mandato IF condicional y a continuación bifurca a FINAL, saltándose el código intermedio. No vuelve al segundo mandato IF. En FINAL, como la prueba de **&C=5** falla, no se llama a PROGA. A continuación, el procedimiento procesa el siguiente mandato, ENDPGM, que indica el final del procedimiento, y devuelve el control al procedimiento que efectuó la llamada.

La lógica del proceso sería diferente si, utilizando el mismo código, los valores iniciales de las variables fueran diferentes. Por ejemplo, si al comienzo de este código el valor de &A es 3 y el valor de &C es 4, la primera sentencia IF se evalúa como falsa. En lugar de procesar el mandato GOTO FINAL, el procedimiento pasa por alto la primera sentencia IF y se desplaza a la siguiente. La segunda sentencia IF se evalúa como verdadera y el valor de &C pasa a ser 5. Las sentencias posteriores, no mostradas aquí, también se procesan consecutivamente. Cuando el proceso alcanza la última sentencia IF, la condición **&C=5** se evalúa como verdadera y se llama a PROGA.

Se ejecutan una serie de sentencias IF de forma independiente. Por ejemplo:

```
        PGM /*  IFFY  */
        DCL &A...
        DCL &B...
        DCL &C...
        DCL &D...
        DCL &AREA *CHAR LEN(5) VALUE(YESNO)
        DCL &RESP...
        IF (&A=&B) THEN(GOTO END)   /*  SI #1  */
        IF (&C=&D) THEN(CALL PGMA)   /*  SI #2  */
```

```
        IF (&RESP=1) THEN(CHGVAR &C 2)  /*  SI #3  */
        IF (%SUBSTRING(&AREA 1 3) *EQ YES) THEN(CALL PGMB)  /*  SI #4  */
        CHGVAR &B &C
        .
        .
        .
  END:  ENDPGM
```

Si, en este ejemplo, &A no es igual a &B, se ejecuta el próximo mandato. Si &C es igual a &D, se llama a PGMA. Cuando se vuelve de PGMA, se considera la tercera sentencia IF, y así sucesivamente. Observe la diferencia en la lógica y en el proceso entre estas sentencias IF simples secuenciales y la utilización de IF con ELSE, o la utilización de mandatos IF incluidos, descritos más adelante en este capítulo (véanse los apartados "Utilización del Mandato ELSE" en el tema 2.5.4 y "Utilización de Mandatos IF Incluidos" en el tema 2.5.5).

Un mandato incluido es un mandato que está contenido totalmente en el parámetro de otro mandato. En los ejemplos siguientes, los mandatos CHGVAR y DO están incluidos:

```
  IF (&A *EQ &B) THEN(CHGVAR &A (&A+1))

  IF (&B *EQ &C) THEN(DO)
                      .
                      .
                      .
                      ENDDO
```

2.5.3 Utilización del mandato DO y de grupos DO

El mandato DO le permite procesar juntos un grupo de mandatos. El grupo se define como todos los mandatos que se encuentran entre el mandato DO y el siguiente mandato ENDDO.

El proceso del grupo normalmente está condicionado a la evaluación de un mandato asociado. Los grupos DO se asocian muy frecuentemente con los mandatos IF, ELSE o MONMSG. Por ejemplo:

IMAGEN 2

Si la expresión lógica **(&A=&B)** es cierta, se procesa el grupo Do. Si la expresión no es cierta, el proceso se inicia después del mandato ENDDO; el grupo Do se salta.

En el siguiente procedimiento, si &A no es igual a &B, el sistema llama a PROCB. No se llama a PROCA, ni se procesan los otros mandatos del grupo DO.

IMAGEN 3

Los grupos Do pueden anidarse dentro de otros grupos Do, hasta un máximo de 10 niveles de jerarquización.

Existen tres niveles de jerarquización en el ejemplo siguiente. Observe cómo cada grupo Do finaliza con un mandato ENDDO.

IMAGEN 4

En este ejemplo, si &A en la primera jerarquía no es igual a 5, se llama a PGMC. Si &A no es igual a 5, las sentencias del segundo grupo DO no se procesan. Si &AREA en el segundo grupo DO no es igual a YES, se llama al procedimiento ACCTSPAY, debido a que el proceso se traslada al mandato siguiente después del grupo DO.

El compilador CL no indica el inicio ni el final de los grupos Do. Si éste detecta condiciones no compensadas como errores, no resulta fácil detectar los errores reales. Consulte el mandato DSPCLPDO en QUSRTOOL para ver un ejemplo de cómo indicar la jerarquización de grupos Do.

2.5.4 Utilización del Mandato ELSE

El mandato ELSE es una forma de especificar un proceso alternativo si la condición en el mandato IF asociado es falsa.

El mandato IF puede utilizarse sin el mandato ELSE:

```
IF (&A=&B) THEN(CALLPRC PROCA)
CALLPRC PROCB
```

En este caso, se llama a PROCA solamente si **&A=&B**, pero siempre se llama a PROCB.

Sin embargo si utiliza un mandato ELSE en este procedimiento, la lógica del proceso cambia. En el siguiente ejemplo, si **&A=&B**, se llama a PROCA, y no se llama a PROCB. Si la expresión **&A=&B** no es cierta, se llama a PROCB.

```
IF (&A=&B) THEN(CALLPRC PROCA)
ELSE CMD(CALLPRC PROCB)
CHGVAR &C 8
```

El mandato ELSE debe utilizarse cuando una evaluación falsa de una expresión IF conduce a una bifurcación distinta (es decir, una exclusiva/bifurcación.)

La utilidad real del mandato ELSE se demuestra mejor cuando se combina con grupos Do. En el ejemplo siguiente, el grupo Do puede no ejecutarse, dependiendo de la evaluación de la expresión IF, pero los mandatos restantes siempre se procesan.

IMAGEN 5

Con el mandato ELSE se puede especificar que un mandato o conjunto de mandatos se procesen solamente si la expresión no es verdadera, completando así las alternativas lógicas:

IMAGEN 6

Cada mandato ELSE debe tener un mandato IF asociado que le preceda. Si hay mandatos IF anidados, cada mandato ELSE se empareja con el mandato IF más interior que no se ha emparejado todavía con otro mandato ELSE.

```
IF ... THEN ...
IF ...THEN(DO)
          IF ...THEN(DO)
                     .
                     .
                     .
          ENDDO
          ELSE DO
                     IF ...THEN(DO)
                          .
                          .
                          .
                          ENDDO
                     ELSE DO
                          .
                          .
                          .
                          ENDDO
          ENDDO
ELSE IF ... THEN ...
IF ... THEN ...
IF ... THEN ...
```

Al revisar su procedimiento para ver si hay mandatos ELSE emparejados, empiece siempre por el conjunto más anidado.

El mandato ELSE puede utilizarse para probar una serie de opciones que se excluyen mutuamente. En el ejemplo siguiente, después de la primera prueba satisfactoria de IF, se procesa el mandato incluido y el procedimiento procesa el mandato RCLRSC:

```
IF  COND(&OPTION=1) THEN(CALLPRC PRC(ADDREC))
 ELSE    CMD(IF COND(&OPTION=2) THEN(CALLPRC PRC(DSPFILE)))
  ELSE   CMD(IF COND(&OPTION=3) THEN(CALLPRC PRC(PRINTFILE)))
    ELSE CMD(IF COND(&OPTION=4) THEN(CALLPRC PRC(DUMP)))
```

```
RCLRSC
RETURN
```

2.5.5 Utilización de Mandatos IF Incluidos

Un mandato IF puede estar incluido en otro mandato IF. Esta situación se da cuando el mandato que se va a procesar tras una evaluación verdadera (el mandato CL situado en el parámetro THEN) es, a su vez, otro mandato IF:

```
IF (&A=&B) THEN(IF (&C=&D) THEN(GOTO END))
GOTO START
```

Esto puede resultar útil cuando deben cumplirse varias condiciones para que se ejecute un mandato determinado o un grupo de mandatos. En el ejemplo anterior, si la primera expresión es cierta, el sistema lee el primer parámetro THEN; dentro de él, si la expresión **&C=&D** se evalúa como verdadera, el sistema procesa el mandato del segundo parámetro THEN, GOTO END. Ambas expresiones han de ser ciertas para que se procese el mandato GOTO END. Si una u otra es falsa, se ejecuta el mandato GOTO START. Nótese el uso de paréntesis para organizar expresiones y mandatos.

En programación CL se permiten un máximo de 10 niveles de inclusión de este tipo.

A medida que los niveles de inclusión aumentan y la lógica crece en complejidad, tal vez desee entrar el código con diseño de formato libre para clarificar las relaciones:

```
PGM
DCL &A *DEC 1
DCL &B *CHAR 2
DCL &RESP *DEC 1
IF (&RESP=1) +
    IF (&A=5) +
          IF (&B=NO) THEN(DO)
                      .
                      .
                      .
                      ENDDO
CHGVAR &A VALUE(8)
CALL PGM(DAILY)
ENDPGM
```

La serie IF anterior se trata como un mandato incluido. Cuando una de las condiciones IF falla, el proceso se bifurca al resto del código (CHGVAR y los mandatos posteriores). Si el propósito de este código es acumular una serie de condiciones, que deben ser todas ciertas para que se procese el grupo Do, podría codificarse más fácilmente utilizando *AND con varias expresiones en un solo mandato. Véase el apartado "Utilización de los Operadores *AND, *OR y *NOT" en el tema 2.5.6.

En algunos casos, sin embargo, la bifurcación debe ser diferente dependiendo de la condición que falle. Puede llevar esto a cabo añadiendo un mandato ELSE a cada mandato IF incluido:

```
PGM
DCL &A ...
DCL &B ...
DCL &RESP ...
IF (&RESP=1) +
      IF (&A=5) +
            IF (&B=NO) THEN(DO)
                        .
                        .
                        .
                        SNDPGMMSG ...
                        .
                        .
                        .
                       ENDDO
            ELSE CALLPRC PROCA
      ELSE CALLPRC PROCB
CHGVAR &A 8
CALLPRC PROC(DAILY)
ENDPGM
```

Aquí, si todas las condiciones son ciertas, el mandato SNDPGMMSG se procesa seguido por el mandato CHGVAR. Si la primera y la segunda condición son verdaderas (**&RESP=1** y **&A=5**) pero la tercera (**&B=NO**) es falsa, se llama a PROCA; cuando PROCA vuelve, se procesa el mandato CHGVAR. Si la segunda condición falla, se llama a PROCB (**&B=NO** no se prueba), seguido del mandato CHGVAR.Finalmente, si &RESP no es igual a 1, se procesa inmediatamente el mandao CHGVAR. El mandato ELSE se ha utilizado para proporcionar una bifurcación distinta en cada prueba.

Nota: Los tres ejemplos siguientes son equivalentes sintácticamente

correctos al mandato IF incluido del ejemplo anterior:

```
IF (&RESP=1) THEN(IF (&A=5) THEN(IF (&B=NO) THEN(DO)))

IF (&RESP=1) THEN +
            (IF (&A=5) THEN +
                          (IF (&B=NO) THEN(DO)))

IF (&RESP=1) +
         (IF (&A=5) +
               (IF (&B=NO) THEN(DO)))
```

*2.5.6 Utilización de los Operadores *AND, *OR y *NOT*

*AND y *OR son los valores reservados de los operadores lógicos utilizados para especificar la relación entre los operandos de una expresión lógica. El símbolo & puede sustituir al valor reservado *AND y la barra vertical (|) a *OR. Los valores reservados deben estar precedidos y seguidos de espacios en blanco. Los operandos de una expresión lógica consisten en expresiones, variables lógicas o constantes lógicas separadas por operadores lógicos. El operador *AND indica que los dos operandos (a ambos lados del operador) han de ser ciertos para dar un resultado verdadero. El operador *OR indica que uno de los operandos debe ser cierto para dar un resultado verdadero.

En las expresiones se utilizan otros operadores que no son lógicos para indicar la acción que se va a realizar con los operandos de la expresión o la relación entre los operandos. Existen tres tipos de operadores que no son lógicos:

- aritméticos (+, -, *, /)
- de caracteres (*CAT, ||, *BCAT, |>, *TCAT, |<)
- relacional (*EQ, =, *GT, >, *LT, <, *GE, >=, *LE, <=, *NE, ¬=, *NG, ¬>, *NL, ¬<)

Puede obtenerse información sobre estos operadores en la publicación CL Reference.

A continuación se facilitan ejemplos de expresiones lógicas:

```
((&C *LT 1) *AND (&TIME *GT 1430))
(&C *LT 1 *AND &TIME *GT 1430)
((&C < 1) & (&TIME>1430))
((&C<1) & (&TIME>1430))
```

En cada uno de estos casos, la expresión lógica consta de tres partes: dos operandos y un operador (*AND u *OR, o sus símbolos). Es el tipo de operador (*AND u *OR) el que caracteriza la expresión como lógica, no el tipo de operando. Los operandos de las expresiones lógicas pueden ser variables lógicas u otras expresiones, tales como expresiones relacionales (caracterizadas mediante los símbolos >, <, ó = ó valores reservados correspondientes). Así pues, en el ejemplo:

```
((&C *LT 1) *AND (&TIME *GT 1430))
```

toda la expresión lógica está entre paréntesis y los dos operandos son expresiones relacionales, también encerradas entre paréntesis. Como puede ver en el segundo ejemplo de expresiones lógicas, no es necesario que los operandos estén encerrados entre otros paréntesis, pero es conveniente para una mayor claridad. Los paréntesis no son necesarios porque *AND y *OR tiene distinta prioridad. *AND siempre se considera antes que *OR. En el caso de operadores que tienen la misma prioridad, se pueden utilizar paréntesis para controlar el orden en que se llevan a cabo las operaciones.

Una expresión relacional simple se puede escribir como la condición en un mandato:

```
IF (&A=&B) THEN(DO)
              .
              .
              .
              ENDDO
```

Los operandos de esta expresión relacional también podrían ser constantes.

Si desea especificar más de una condición, puede utilizar una expresión lógica con expresiones relacionales como operandos:

```
IF ((&A=&B) *AND (&C=&D)) THEN(DO)
                          .
                          .
                          .
                          ENDDO
```

La serie de mandatos IF dependientes citados como ejemplo en "Utilización de Mandatos IF Incluidos" en el tema 2.5.5 podría codificarse así:

```
PGM
DCL &RESP *DEC 1
DCL &A *DEC 1
DCL &B *CHAR 2
IF ((&RESP=1) *AND (&A=5) *AND (&B=NO)) THEN(DO)
                                        .
                                        .
                                        .
```

```
ENDDO
  CHGVAR &A VALUE(8)
  CALLPRC PROC(DAILY)
  ENDPGM
```

Aquí los operadores lógicos se utilizan de nuevo entre expresiones relacionales.

Puesto que una expresión lógica también puede tener otras expresiones lógicas como operandos, es posible llegar a una lógica bastante compleja:

```
  IF (((&A=&B) *OR (&A=&C)) *AND ((&C=1) *OR (&D='0'))) THEN(DO)
```

En este caso, &D se define como una variable lógica.

El resultado de la evaluación de cualquier expresión relacional o lógica es '1' ó '0' (verdadero o falso). El mandato dependiente sólo se procesa si la expresión completa se evalúa como verdadera ('1'). El siguiente mandato se interpreta en estos términos:

```
  IF   ((&A = &B) *AND (&C = &D)) THEN(DO)

       ((verdadera'1') *AND (no verdadera'0'))
            (no verdadera '0')
```

La expresión se evalúa finalmente, como no verdadera ('0'); por lo tanto, el mandato DO no se procesa. Para obtener una explicación de cómo se llega a esta evaluación véase las matrices posteriormente en este apartado.

Este mismo proceso se utiliza para evaluar una expresión lógica que utiliza variables lógicas, como en este ejemplo:

```
  PGM
  DCL &A *LGL
  DCL &B *LGL
  IF (&A *OR &B) THEN(CALL PGM(PGMA))
  .
  .
  .
  ENDPGM
```

Aquí la expresión condicional se evalúa para ver si el valor de &A o &B es igual a '1' (cierto). Si uno u otro es verdadero, toda la expresión es verdadera, y se llama a PGMA.

La evaluación final conseguida para todos estos ejemplos de expresiones lógicas se basa en las matrices estándares que comparan dos valores (denominadas aquí &A y &B) bajo un operador *OR o *AND.

Utilice la siguiente matriz al utilizar *OR con constantes o variables lógicas:

```
Si &A es:         '0'  '0'  '1'  '1'
y &B es:          '0'  '1'  '0'  '1'

o la expresión OR es: '0'  '1'  '1'  '1'
```

En resumen, cuando hay varios operadores OR con constantes o variables lógicas, la expresión es falsa ('0') si todos los valores son falsos. La expresión es verdadera ('1') si algún valor es verdadero.

```
  PGM
  DCL &A *LGL VALUE('0')
  DCL &B *LGL VALUE('1')
  DCL &C *LGL VALUE('1')
  IF (&A *OR &B *OR &C) THEN(CALL PGMA)
  .
  .
  .
  ENDPGM
```

Aquí los valores no son todos falsos; por lo tanto la expresión es verdadera y se llama a PGMA.

Utilice la matriz siguiente al evaluar una expresión lógica con *AND con constantes o variables lógicas:

```
Si &A es:         '0'  '0'  '1'  '1'
y  &B es:         '0'  '1'  '0'  '1'

la expresión AND es: '0'  '0'  '0'  '1'
```

En el caso de varios operadores AND con constantes o variables lógicas, la

expresión es falsa ('0') si algún valor es falso y es verdadera cuando todos los valores son verdaderos.

```
  PGM
  DCL &A *LGL VALUE('0')
  DCL &B *LGL VALUE('1')
  DCL &C *LGL VALUE('1')
  IF (&A *AND &B *AND &C) THEN(CALL PGMA)
  .
  .
  .
  ENDPGM
```

Los valores no son todos ciertos; por lo tanto, la expresión es falsa y no se llama a PGMA.

Estos operadores lógicos pueden utilizarse solamente *en* una expresión donde los operandos representen un valor lógico, como en los ejemplos anteriores. Es *incorrecto* utilizar OR o AND para variables que no son lógicas. Por ejemplo:

```
  PGM
  DCL &A *CHAR 3
  DCL &B *CHAR 3
  DCL &C *CHAR 3

  Incorrecto:   IF (&A *OR &B *OR &C = YES) THEN...
```

La codificación correcta sería:

```
  IF ((&A=YES) *OR (&B=YES) *OR (&C=YES)) THEN...
```

En este caso, el operador OR se encuentra entre expresiones relacionales.

El operador lógico *NOT (o ¬) se utiliza para negar variables o constantes lógicas. Cualquier operador *NOT debe evaluarse antes de que se evalúen los operadores *AND u *OR. Cualquier valor que siga a los operadores *NOT debe evaluarse antes de que la relación lógica entre los operandos se evalúe.

```
  PGM
  DCL &A *LGL '1'
  DCL &B *LGL '0'
  IF (&A *AND *NOT &B) THEN(CALL PGMA)
```

En este ejemplo, todos los valores son verdaderos; por consiguiente, la expresión es verdadera y se llama a PGMA.

```
  PGM
  DCL &A *LGL
  DCL &B *CHAR 3 VALUE('ABC')
  DCL &C *CHAR 3 VALUE('XYZ')
  CHGVAR &A VALUE(&B *EQ &C)
  IF (&A) THEN(CALLPRC PROCA)
```

En este ejemplo, el valor es falso, y por consiguiente, no se llama a PROCA.

Para obtener más información sobre las expresiones lógicas y relacionales, véase la publicación CL Reference.

2.5.7 Utilización de la Función Incorporada %BINARY

La función incorporada binaria (%BINARY o %BIN) interpreta el contenido de una variable de tipo carácter CL especificada como un entero binario con signo. La posición inicial empieza en la posición especificada y tiene una longitud de 2 ó 4 caracteres.

La sintaxis de la función binaria incorporada es:

```
  %BINARY(nombre-variable-tipo-carácter posición-inicial longitud)
```

o

```
  %BIN(nombre-variable-tipo-carácter posición-inicial longitud)
```

La posición inicial y la longitud son opcionales; sin embargo, si la posición inicial y la longitud *no están especificadas*, se utiliza una posición inicial de 1 y la longitud de la variable de tipo carácter especificada. La longitud de la variable de tipo carácter debe declarase como 2 ó 4.

Si la posición inicial *está especificada*, también debe especificar una longitud constante de 2 ó 4. La posición inicial ha de ser un número positivo igual o mayor que 1. Si la suma de la posición inicial y la longitud es mayor que la longitud de la variable de tipo carácter, se produce un error. (También puede utilizarse una variable decimal CL para la posición inicial.)

Puede utilizar la función incorporada binaria con los mandatos IF y CHGVAR. Puede usarse como tal o como parte de una expresión aritmética o lógica. También puede utilizar la función incorporada binaria en cualquier parámetro de mandato que este definido como numérico (TYPE de *DEC, *INT2 ó *INT4) con EXPR(*YES).

Cuando la función incorporada binaria se utiliza con el parámetro condición (COND) en el mandato IF o con el parámetro VALUE en el mandato Cambiar Variable (CHGVAR), el contenido de la variable de tipo carácter se interpreta como una conversión binaria-a-decimal.

Cuando la función incorporada binaria se utiliza con el parámetro VAR en el mandato CHVAR, el valor decimal en el parámetro VALUE se convierte a entero binario con signo de 2 bytes o 4 bytes y el resultado se almacena en una variable de tipo carácter en la posición inicial especificada. Las fracciones decimales se truncan.

La función incorporada binaria se utiliza con el parámetro RTNVAL del mandato CALLPRC para indicar que el procedimiento que efectúa la llamada espera que el procedimiento llamado devuelva un entero binario.

Una variable de tipo carácter de 2 bytes puede contener valores enteros binarios con signo desde -32.768 hasta 32.767. Una variable de tipo carácter de 4 bytes puede contener valores enteros binarios con signo desde -2.147.483.648 hasta 2.147.483.647.

A continuación se facilitan ejemplos de la función binaria incorporada:

□

```
  DCL    VAR(&B2)  TYPE(*CHAR)  LEN(2)    VALUE(X'001C')
  DCL    VAR(&N)   TYPE(*DEC)   LEN(3 0)
  CHGVAR   &N   %BINARY(&B2)
```

El contenido de la variable &B2 se trata como un entero binario con signo de 2 bytes y se convierte a su equivalente decimal, que es 28. Después se asigna a la variable decimal &N.

□

```
  DCL    VAR(&N)   TYPE(*DEC)   LEN(5 0)  VALUE(107)
  DCL    VAR(&B4)  TYPE(*CHAR)  LEN(4)
  CHGVAR   %BIN(&B4)    &N
```

El valor de la variable decimal &N se convierte a un número binario con signo de 4 bytes, y se sitúa en la variable tipo carácter &B4. La variable &B4 contendrá el valor de X'0000006B'.

□

```
  DCL    VAR(&P)  TYPE(*CHAR)  LEN(100)
  DCL    VAR(&L)  TYPE(*DEC)   LEN(5 0)
  CHGVAR   &L  VALUE(%BIN(&P 1 2) * 5)
```

Los dos primeros caracteres de la variable &P se tratan como enteros binarios con signo, se convierten al decimal equivalente, y se

multiplican por 5. El producto se asigna a la variable decimal &L.

```
  DCL    VAR(&X)   TYPE(*CHAR)  LEN(50)
  CHGVAR   %BINARY(&X 15 2)  VALUE(122,56)
```

El número 122,56 se trunca al número entero 122, se convierte a entero binario con signo de 2 bytes y se coloca en las posiciones 15 y 16 de la variable de tipo carácter &X. Estas posiciones contendrán el equivalente hexadecimal X'007A'.

```
  DCL    VAR(&B4)  TYPE(*CHAR)  LEN(4)
  CHGVAR   %BIN(&B4)  VALUE(-57)
```

El valor -57 se convierte a entero binario con signo de 4 bytes y se asigna a la variable de tipo carácter &B4., que contendrá el valor X'FFFFFFC7'.

```
  DCL    VAR(&B2)  TYPE(*CHAR)  LEN(2)   VALUE(X'FF1B')
  DCL    VAR(&C5)  TYPE(*CHAR)  LEN(5)
  CHGVAR   &C5  %BINARY(&B2)
```

El contenido de la variable &B2 se trata como un entero binario con signo de 2 bytes y se convierte a su equivalente decimal, que es -229. Este número se convierte a formato carácter y se almacena en la variable de tipo carácter &C5. Esta variable contendrá el valor '-0229'.

```
  DCL    VAR(&C5)  TYPE(*CHAR)  LEN(5)   VALUE(' 1253')
  DCL    VAR(&B2)  TYPE(*CHAR)  LEN(2)
  CHGVAR   %BINARY(&B2)  VALUE(&C5)
```

El número de carácter 1253 en la variable tipo carácter &C5 se convierte a un número decimal .El número decimal 1253 se convierte a entero binario con signo de 2 bytes y se almacena en la variable &B2. Ésta tendrá el valor X'04E5'.

```
  DCL    VAR(&S)  TYPE(*CHAR)  LEN(100)
  IF     (%BIN(&S 1 2) > 10)
         THEN( SNDPGMMSG MSG('Demasiados en lista.') )
```

Los 2 primeros bytes de la variable de tipo carácter &S se tratan como un entero binario con signo cuando se comparan con el número 10. Si el número binario tiene un valor mayor que 10, se ejecuta el mandato SNDPGMMSG (Enviar Mensaje a Programa).

```
  DCL    VAR(&RTNV) TYPE(*CHAR) LEN(4)
  CALLPRC PRC(PROCA) RTNVAL(%BIN(&RTNV 1 4))
```

El procedimiento PROCA devuelve un entero de 4 bytes que se almacena en la variable &RTNV.

2.5.8 Utilización de la Función Incorporada %SUBSTRING

La función incorporada de subseries (%SUBSTRING ó %SST) produce una serie de caracteres que es un subconjunto de una serie de caracteres existente y que sólo puede utilizarse en un procedimiento CL. En un mandato CHGVAR, la función %SST puede especificarse en vez de la variable (parámetro VAR) que ha de cambiarse o del valor (parámetro VALUE) por el que la variable va a cambiarse. En un mandato IF, la función %SST puede especificarse en la expresión.

El formato de la función incorporada de subseries es:

```
%SUBSTRING(nombre-variable-tipo-carácter posición-inicial longitud)
```

o

```
%SST(nombre-variable-tipo-carácter posición-inicial longitud)
```

Puede codificar *LDA en lugar del nombre de variable de tipo carácter para indicar que la función de subseries se realiza en el contenido del área de datos local.

La función de subseries produce una subserie del contenido de una variable CL de tipo carácter especificada o del área de datos local. La subserie empieza en la posición inicial especificada (que puede ser un nombre de variable) y tiene la longitud especificada (que también puede ser un nombre de variable). Ninguno de los dos valores, ni la posición inicial ni la longitud, pueden ser 0 o negativos. Si la suma de la posición inicial y la longitud de la subserie es mayor que la longitud de la variable o del área de datos local, se produce un error. La longitud del área local de datos es 1024.

A continuación se facilitan ejemplos de la función incorporada de subseries:

- Si las primeras dos posiciones de la variable de tipo carácter &NAME son IN, se llama al programa INV210. El valor entero de &NAME se pasa a INV210 y el valor de &ERRCODE no se cambia. De lo contrario, el valor de %ERRCODE se establece en 99.

```
DCL &NAME *CHAR VALUE(INVOICE)
DCL &ERRCODE *DEC (2 0)
IF (%SST(&NAME 1 2) *EQ 'IN') +
THEN(CALL INV210 &NAME)
ELSE CHGVAR &ERRCODE 99
```

- Si las primeras dos posiciones de &A coinciden con las dos primeras posiciones de &B, se llama al programa CUS210.

```
DCL &A *CHAR VALUE(ABC)
DCL &B *CHAR VALUE(DEF)
IF (%SST(&A 1 2) *EQ %SUBSTRING(&B 1 2)) +
CALL CUS210
```

- La posición y la longitud también pueden ser variables. En este ejemplo se cambia el valor de &X que comienza en la posición &Y para la longitud &Z por 123.

```
CHGVAR %SST(&X &Y &Z) '123'
```

- Si &A es **ABCDEFG** antes de ejecutar este mandato CHGVAR, &A es

```
CHGVAR %SST(&A 2 3) '123'
```

A123EFG después de ejecutarlo.

- En este ejemplo, la longitud de la subserie, 5, excede la longitud del operando YES con el que se compara. El operando se rellena con espacios en blanco para que la comparación se efectúe entre YESNO y YESbb (siendo b un espacio en blanco). La condición es falsa.

```
DCL VAR(&NAME) TYPE(*CHAR) LEN(5) VALUE(YESNO)
.
.
.
IF (%SST (&NAME 1 5) *EQ YES) +
  THEN(CALL PROGA)
```

Si la longitud de la subserie es menor que la del operando, la subserie se rellena con espacios en blanco para la comparación. Por ejemplo:

```
DCL VAR(&NAME) TYPE(*CHAR) LEN(5) VALUE(YESNO)
.
```

```
      .
      .
      IF (%SST(&NAME 1 3 ) *EQ YESNO) THEN(CALL PROG)
```

Esta condición es falsa porque YESbb (siendo bb dos espacios en blanco) no es igual a YESNO.

- El valor de la variable &A se coloca en las posiciones 1 a 10 del área de datos local.

```
  CHGVAR %SST(*LDA 1 10) &A
```

- Si la concatenación de las posiciones 1 a 3 del área de datos local más la constante 'XYZ' es igual a la variable &A, entonces se llama a PROCA.Por ejemplo, si las posiciones 1 a 3 del área de datos local contiene 'ABC', y la variable &A tiene un valor **ABCXYZ**, la prueba da cierto como resultado y se llama a PROCA.

```
  IF (((%SST*LDA 1 3) *CAT 'XYZ') *EQ &A) THEN(CALLPRC PROCA)
```

- Este procedimiento explora la variable de tipo carácter &NUMBER y cambia todos los ceros iniciales por blancos. Esto puede utilizarse para la edición de un campo antes de visualizarlo en un mensaje.

```
        DCL &NUMBER *CHAR LEN(5)
        DCL &X *DEC LEN(3 0) VALUE(1)
        .
        .
  LOOP:IF (%SST(&NUMBER &X 1) *EQ '0') DO
        CHGVAR (%SST(&NUMBER &X 1)) ' '  /* Blanco fuera */
        CHGVAR &X (&X + 1)  /* Incrementa */
        IF (&X *NE 5) GOTO LOOP
        ENDDO
```

El siguiente procedimiento utiliza la función incorporada de subserie para buscar la primera sentencia en un campo de 50 caracteres &INPUT y para situar el texto restante en un campo &REMAINDER. Se presupone que una sentencia debe tener por lo menos 2 caracteres sin ningún punto intercalado.

```
        PGM (&INPUT &REMAINDER)   /*  BUSCA  */
        DCL &INPUT *CHAR LEN(50)
        DCL &REMAINDER *CHAR LEN(50)
        DCL &X *DEC LEN(2 0) VALUE(03)
        DCL &L *DEC LEN(2 0)   /*  LONGITUD RESTANTE */
SCAN:   IF (%SST(&INPUT &X 1) *EQ '.') THEN(DO)
            CHGVAR VAR(&L) VALUE(50-&X)
            CHGVAR VAR(&X) VALUE(&X+1)
            CHGVAR VAR(&REMAINDER) VALUE(%SST(&INPUT &X &L))
            RETURN
            ENDDO
        IF (&X *EQ 49) THEN(RETURN)
        CHGVAR &X (&X+1)
        GOTO SCAN
        ENDPGM
```

El procedimiento empieza buscando un punto en la tercera posición. Observe que la función de subseries comprueba &INPUT desde la posición 3 con una longitud de 1, que es la posición 3 solamente (la longitud no puede ser cero). Si la posición 3 es un punto, se calcula la longitud restante de &INPUT. El valor de &X se avanza hasta el principio del resto y la parte restante de &INTPUT se mueve a &REMAINDER.

Si la posición 3 no es un punto, el procedimiento lo busca en la posición 49. Si es así, se supone que la posición 50 es un punto y vuelve. Si no está en la posición 49, el proceso avanza &X a la posición 4, y repite el proceso.

2.5.9 Utilización de la Función Incorporada %SWITCH

La función incorporada conmutar (%SWITCH) compara uno o más de ocho conmutadores con los ocho valores de conmutadores ya establecidos para el trabajo y devuelve un valor lógico de '0' ó '1'. Los valores iniciales de los conmutadores para el trabajo se determinan primero con el mandato Crear Descripción de Trabajo (CRTJOBD); el valor por omisión es **00000000.** Puede cambiarlo si es necesario utilizando el parámetro SWS en los mandatos SBMJOB, CHGJOB o JOB; el valor por omisión es el valor de descripción del trabajo. Otros lenguajes de alto nivel también pueden asignar conmutadores de trabajo.

Si, en la comparación de los valores %SWITCH con los valores del trabajo, cada conmutador es igual, se devuelve un valor lógico de '1' (verdadero). Si algún conmutador probado no tiene el valor indicado, el resultado es un '0' (falso).

La sintaxis de la función incorporada %SWITCH es:

```
  %SWITCH(máscara-8-caracteres)
```

La máscara de 8 caracteres se utiliza para indicar qué conmutadores de trabajo van a probarse y qué valor ha de probarse para cada conmutador. Cada posición de la máscara corresponde con uno de los ocho conmutadores de un trabajo. La posición 1 corresponde al conmutador de trabajo 1, la posición 2 al conmutador 2, etc. Cada posición de la máscara puede especificarse como uno de estos tres valores: 0, 1 ó X.

0 Se va a probar el conmutador de trabajo correspondiente para ver si es 0 (desactivado).

1 Se va a probar el conmutador de trabajo correspondiente para ver si es 1 (activado).

X El conmutador de trabajo correspondiente no va a probar. El valor del conmutador no afecta al resultado de %SWITCH.

Si se especifica **%SWITCH(0X111XX0),** los conmutadores de trabajo 1 y 8 se prueban para ver si son 0; los conmutadores 3, 4 y 5 se prueban para ver si son 1; y los conmutadores 2, 6 y 7 no se prueban. Si cada conmutador de trabajo contiene el valor (1 ó 0 solamente) mostrado en la máscara, el resultado de %SWITCH es verdadero ('1').

Los conmutadores pueden probarse en un CL para controlar el flujo del procedimiento. Esta función se utiliza en procedimientos CL con los mandatos IF y CHGVAR. Los conmutadores pueden cambiarse en un procedimiento CL mediante el mandato Cambiar Trabajo (CHGJOB). Para procedimientos CL, estos cambios entran en vigor inmediatamente.

Subtemas

2.5.9.1 %SWITCH con el Mandato IF

En el mandato IF, se puede especificar %SWITCH en el parámetro COND como la expresión lógica que se ha de probar. En el siguiente ejemplo, **0X111XX0** se compara con el valor del conmutador de trabajo predeterminado:

```
  IF  COND(%SWITCH(0X111XX0))  THEN(GOTO C)
```

Si los conmutadores 1, 2, 4, 5, y 8 contienen 0, 1, 1, 1, y 0, respectivamente, el resultado es verdadero y el procedimiento se bifurca hacia el mandato que tenga la etiqueta C. Si uno o más de los conmutadores probados no tienen los valores indicados en la máscara, el resultado es falso y no se produce la bifurcación.

En el ejemplo siguiente, los conmutadores controlan el proceso condicional de dos procedimientos.

```
  SBMJOB  JOB(APP502) JOBD(PAYROLL) CMD(CALL APP502)
      SWS(11000000)

  PGM  /* CONTROL */
  IF (%SWITCH(11XXXXXX)) CALLPRC PROCA
  IF (%SWITCH(10XXXXXX)) CALLPRC PROCB
  IF (%SWITCH(01XXXXXX)) CALLPRC PROCC
  IF (%SWITCH(00XXXXXX)) CALLPRC PROCD
  ENDPGM

  PGM  /* PROCA */
  CALLPRC TRANS
  IF (%SWITCH(1XXXXXXX)) CALLPRC CUS520
  ELSE CALLPRC CUS521
  ENDPGM
```

2.5.9.2 %SWITCH con el Mandato CHGVAR

En el mandato CHGVAR, puede especificar %SWITCH para cambiar el valor de una variable lógica. El valor de dicha variable está determinado por el resultado de la comparación entre sus valores de %SWITCH con los valores de los conmutadores de trabajo. Si el resultado de la comparación es verdadero, la variable lógica se establece en '1'. Si el resultado es falso, se establece en '0'. Por ejemplo, si el conmutador de trabajo se establece en **10000001** y se procesa este programa:

```
  PGM
  DCL &A *LGL
  CHGVAR VAR(&A) VALUE(%SWITCH(10000001))
  .
  .
  .
  ENDPGM
```

entonces la variable &A contiene el valor '1'.

2.5.10 Utilización del Mandato Supervisar Mensaje (MONMSG)

Los mensajes de escape se envían a los procedimientos CL por los mandatos de los procedimientos CL y los programas y procedimientos que estos llaman. Estos mensajes se envían para indicar los procedimientos en los que se han detectado errores y que no han realizado las funciones solicitadas. Los procedimientos CL pueden supervisar la recepción de mensajes de escape, y usted puede especificar con mandatos cómo se deben manejar los mensajes. Por ejemplo, si un procedimiento CL intenta mover un área de datos que se ha suprimido, se envía un mensaje de escape que indica que el objeto no se ha encontrado al procedimiento del mandato Mover Objeto (MOVOBJ).

Utilizando el mandato Supervisar Mensaje (MONMSG), puede indicar a un procedimiento que tome una acción predeterminada si se producen errores específicos durante el proceso del mandato inmediatamente anterior. El mandato MONMSG se utiliza para supervisar los mensajes de escape, de
|notificación o de estado enviados a la pila de llamada del procedimiento en el cual se utiliza el mandato. El mandato MONMSG tiene los siguientes parámetros:

```
  MONMSG   MSGID(identificador-de-mensajes) CMPDTA(datos-de-comparación) +
           EXEC(mandato-CL)
```

Cada mensaje enviado a causa de un error tiene un identificador exclusivo. Puede entrar 50 identificadores como máximo en el parámetro MSGID (Consulte el manual CL Reference para ver los mensajes e identificadores). El parámetro CMPDTA permite especificar con más detalle los mensajes de error, ya que puede comprobar si en la parte MSGDTA del mensaje hay una serie de caracteres determinada. En el parámetro EXEC, puede especficificar un mandato CL (como CALL, DO, o GOTO), que indique al procedimiento que efectúe la recuperación de errores.

|En el siguiente ejemplo, el mandato MONMSG sigue al mandato Recibir Archivo (RCVF) y, por lo tanto, solo supervisa los mensajes enviados por el mandato RCVF.

```
|  READLOOP:  RCVF                          /* Leer un registro de archivo */
|             MONMSG  MSGID(CPF0864)  EXEC(GOTO CMDLBL(EOF))
|             /*  Procesa el registro de archivo */
|             GOTO  CMDLBL(READLOOP)       /*  Obtener otro registro */
|  EOF:       /*  Proceso fin de archivo  */
```

|El mensaje de escape, CPF0864, se envía a la cola de invocación del |procedimiento cuando no hay mas registros para leer en el archivo. Debido |a que el ejemplo especifica MSGID(CPF0864), MONMSG supervisa esta |condición. Cuando recibe el mensaje, se ejecuta el mandato GOTO.

El usuario también puede utilizar el mandato MOBMSG para supervisar los mensajes enviados por cualquier mandato en un procedimiento CL. El siguiente ejemplo contiene dos mandatos MONMSG. El primero de ellos supervisa los mensajes CPF0001 y CPF1999; estos mensajes puede enviarlos un mandato que se ejecute posteriormente en el procedimiento. Cuando se recibe uno de estos mensajes desde uno de los mandatos que se ejecutan en el procedimiento, el control se bifurca al mandato identificado con la etiqueta EXIT2.

El segundo mandato MONMSG supervisa los mensajes CPF2105 y MCH1211. Debido a que no se codifica ningún mandato para el parámetro EXEC, estos mensajes se ignoran.

```
  PGM
  DCL
  MONMSG MSGID(CPF0001 CPF1999) EXEC(GOTO EXIT2)
  MONMSG MSGID(CPF2105 MCH1211)
  .
  .
  .
  ENDPGM
```

El mensaje CPF0001 indica que se ha detectado un error en el mandato identificado en el propio mensaje. El mensaje CPF1999, que pueden haberlo enviado muchos de los mandatos de depuración, tal como Cambiar Variable de Programa (CHGPGMVAR), indica que se han producido errores en el mandato, pero no identifica en cuál.

Todas las condiciones de error supervisadas por el mandato MONMSG con el parámetro EXEC especificado (CPF0001 o CPF1999) se manejan de la misma forma en EXIT2 y no es posible volver a la siguiente sentencia en secuencia después del error. Esto se puede evitar supervisando las condiciones específicas después de cada mandato y bifurcando hacia procedimientos específicos de corrección de errores.

Todas las condiciones de error supervisadas por el mandato MONMSG sin el

parámetro EXEC especificado (CPF2105 o MCH1211) se ignoran, y el proceso del procedimiento continúa con el siguiente mandato.

Si el error se produce al evaluar la expresión de un mandato IF, la condición se considera falsa. En el ejemplo siguiente, MCH1211 (dividir por cero) podría producirse en el mandato IF. La condición se consideraría falsa y se llamaría a PROCA.

```
IF(&A / &B *EQ 5) THEN(DLTF ABC)
ELSE CALLPRC PROCA
```

Si se codifica el mandato MONMSG al principio del procedimiento CL, los mensajes que se especifiquen se supervisan en todo el programa, sin tener en cuenta que el mandato produce los mensajes. Si se utiliza el parámetro EXEC, sólo se puede especificar el mandato GOTO.

Se puede especificar el mismo identificador de mensaje en un mandato MONMSG a nivel de procedimiento o a nivel de mandato. Los mandatos MONMSG a nivel de mandato tienen prioridad sobre los de nivel de procedimiento. En el ejemplo siguiente, si se recibe un mensaje CPF0001 en CMDB, se ejecuta CMDC. Si se recibe el mensaje CPF0001 en cualquier otro mandato del procedimiento, éste se bifurca hacia EXIT2. Si se recibe un mensaje CPF1999 en cualquier mandato, incluyendo CMBD, el procedimiento se bifurca hacia EXIT2.

```
PGM
MONMSG MSGID(CPF0001 CPF1999) EXEC(GOTO EXIT2)
CMDA
CMDB
MONMSG MSGID(CPF0001) EXEC(CMDC)
CMDD
EXIT2:  ENDPGM
```

Debido a que son muchos los mensajes de escape que pueden enviarse a un procedimiento, debe decidir cuáles desea supervisar y manejar. La mayoría de estos mensajes se envían a un programa solamente si hay un error en el procedimiento. Otros se envían debido a condiciones externas al procedimiento. Habitualmente, un procedimiento CL debe supervisar los mensajes que pertenecen a su función básica y que puede manejar adecuadamente. En el caso de todos los demás mensajes, OS/400 asume que se ha producido un error y toma la acción por omisión apropiada.

Para obtener más información sobre el manejo de mensajes en procedimientos CL, consulte el Capítulo 7 y el Capítulo 8.

2.6 Valores que Pueden Utilizarse como Variables

Subtemas

2.6.1 Recuperación de Valores del Sistema

Un valor de sistema contiene información de control para la operación de ciertas partes del sistema. IBM proporciona varios tipos de valores de sistema. Por ejemplo, QDATE y QTIME son valores de sistema para fecha y hora, que puede fijar cuando se arranca el OS/400.

Puede incluir valores del sistema en el procedimiento y manipularlos como variables utilizando el mandato Recuperar Valor del Sistema (RTVSYSVAL):

```
  RTVSYSVAL  SYSVAL(nombre-valor-sistema)  RTNVAR(nombre-variable-CL)
```

El parámetro RTNVAR especifica el nombre de la variable en el procedimiento CL que va a recibir el valor del valor del sistema.

El tipo de variable debe coincidir con el tipo de valor del sistema. En el caso de valores del sistema lógicos y de tipo carácter, la longitud de la variable CL debe ser igual a la del valor. En el caso de valores decimales, la longitud de la variable debe ser superior o igual a la longitud del valor del sistema. Los atributos de los valores del sistema se definen en el manual *Gestión de Trabajos*.

Subtemas
2.6.1.1 Valor del Sistema QTIME
2.6.1.2 Valor del Sistema QDATE

2.6.1.1 Valor del Sistema QTIME

En el siguiente ejemplo, se ha recibido QTIME y se ha movido a una variable, que después se compara con otra variable.

```
        PGM
        DCL  VAR(&PWRDNTME)  TYPE(*CHAR)  LEN(6)  VALUE('162500')
        DCL  VAR(&TIME)  TYPE(*CHAR)  LEN(6)
        RTVSYSVAL  SYSVAL(QTIME)  RTNVAR(&TIME)
        IF  (&TIME *GT &PWRDNTME)  THEN(DO)
        SNDBRKMSG('Desconexión dentro de 5 min. Finalice la sesión.')
        PWRDWNSYS  OPTION(*CNTRLD)  DELAY(300)  RESTART(*NO)  +
                 IPLSRC(*PANEL)

        ENDDO
        ENDPGM
```

Consulte el manual *Gestión de Trabajos* para obtener una lista de valores del sistema y cómo puede cambiarlos y visualizarlos.

2.6.1.2 Valor del Sistema QDATE

En muchas aplicaciones, tal vez desee utilizar la fecha del día en su procedimiento, recuperando el valor del sistema QDATE y situándolo en una variable. Quizá también desee cambiar el formato de dicha fecha para utilizarla en el procedimiento. Para convertir el formato de una fecha en un programa CL, utilice el mandato Convertir Fecha (CVTDAT).

El formato de la fecha del sistema es el valor del sistema QDATFMT, que inicialmente es MDA (mesdíaaño). Por ejemplo, **062488** es el 24 de junio de 1988 en formato MDA. Puede cambiar este formato por AMD, DMA o JUL (juliano). En el caso del formato juliano, QDAY es un valor de 3 caracteres comprendido entre 001 a 366. Se utiliza para determinar el número de días que hay entre dos fechas. Con el mandato CVTDAT, también puede suprimir los separadores de fecha o cambiar el carácter utilizado como separador.

El formato del mandato CVTDAT es:

```
  CVTDAT    DATE(fecha-a-convertir) TOVAR(variable-CL) +
            FROMFMT(formato-antiguo) TOFMT(nuevo-formato) +
            TOSEP(nuevos-separadores)
```

En el parámetro DATE se puede especificar una constante o una variable a convertir. Una vez que la fecha se ha convertido, se coloca en la variable mencionada en el parámetro TOVAR. En el ejemplo siguiente, la fecha de la variable &DATE, cuyo formato es MDA, se convierte al formato DMA y se coloca en la variable &CVTDAT.

```
  CVTDAT    DATE(&DATE) TOVAR(&CVTDAT) FROMFMT(*MDY) TOFMT(*DMY)
            TOSEP(*SYSVAL)
```

El separador de fechas permanece como se especificó en el valor del sistema QDATSEP.

El mandato CVTDAT puede resultar útil al crear objetos o al añadir un miembro que utiliza una fecha como parte de su nombre. Por ejemplo, suponga que debe añadirse un miembro que utiliza la fecha actual del sistema a un archivo. Suponga también que la fecha actual está en formato MDA y que se va a convertir al formato juliano.

```
  PGM
  DCL &DATE6 *CHAR  LEN(6)
  DCL &DATE5  *CHAR  LEN(5)
  RTVSYSVAL  QDATE  RTNVAR(&DATE6)
  CVTDAT  DATE(&DATE6)  TOVAR(&DATE5)  TOFMT(*JUL)  TOSEP(*NONE)
  ADDPFM  LIB1/FILEX  MBR('MBR'  *CAT  &DATE5)
  .
  .
  .
  ENDPGM
```

Si la fecha actual es el 5 de enero de 1988, el miembro añadido se denominaría MBR88005.

Recuerde lo siguiente al convertir fechas:

- La longitud del valor en el parámetro DATE y la longitud de la variable en el parámetro TOVAR deben ser compatibles con el formato de fecha. La longitud de la variable en el parámetro TOVAR debe ser como mínimo:
  - Para fechas no julianas, de 6 caracteres cuando no se utilizan separadores y de 8 caracteres cuando se utilizan separadores.
  - Para fechas julianas, de 5 caracteres cuando no se utilizan separadores y de 6 caracteres cuando se utilizan separadores.

  Se envían mensajes de error cuando los caracteres convertidos no caben en la variable. Si la fecha convertida es más corta que la variable, ésta se rellena con espacios en blanco por la derecha.
- En todos los formatos de fecha, excepto en el juliano, el año, mes y día son campos de 2 bytes, cualquiera que sea el valor que contiene. Todos los valores convertidos se justifican por la derecha y, si es necesario, se añaden ceros iniciales.
- En el formato juliano, el día es un campo de 3 bytes y el año un campo de 2 bytes. Todos los valores convertidos se justifican por la derecha y, si es necesario, se añaden ceros iniciales.

|Lo siguiente es un programa alternativo que utiliza API enlazable ILE,
|Obtener Hora Local Actual (CEELOCT), para convertir una fecha a formato
|Juliano. Para crear este programa, debe utilizar el mandato CRTBNDCL solo
|o conjuntamente con el mandato CRTPGM.

```
|  PGM
|  DCL  &LILDATE  *CHAR  LEN(4)
```

```
|  DCL  &PICTSTR  *CHAR  LEN(5)  VALUE(YYDDD)
|  DCL  &JULDATE  *CHAR  LEN(5)
|  DCL  &SECONDS  *CHAR  8       /*  Segundos del CEELOCT               */
|  DCL  &GREG     *CHAR  23      /*  Fecha Gregoriana del CEELOCT       */
|                                /*                                     */
|  CALLPRC  PRC(CEELOCT)         /*  Obtener fecha y hora actual        */ +
|           PARMS (&LILDATE)     /*  Fecha en formato Liliano           */ +
|                  &SECONDS      /*  Campo de segundos que no se utiliza */
|                  &GREG         /*  Campo gregoriano que no se utiliza  */
|                  *OMIT         /*  Parámetro de realimentación Omit +
|                                     para que se señalen las excepciones */

|  CALLPRC  PRC(CEEDATE) +
|           PARMS (&LILDATE)     /*  Fecha de hoy                */ +
|                  &PICTSTR      /*  Cómo dar formato            */ +
|                  &JULDATE      /*  Fecha Juliana               */ +
|                  *OMIT

|  ADDPGM   LIB1/FILEX  MBR('MBR'  *CAT   &JULDATE')

|  ENDPGM

|Consulte el manual System API Reference, SC41-4801, para mas información
|acerca de las API ILE.
```

2.6.2 Recuperación del Fuente de Configuración

Utilizando el mandato Recuperar Fuente de Configuración (RTVCFGSRC), puede generar el fuente del mandato CL para crear objetos de configuración existentes y situar el fuente en un miembro de archivo fuente. El fuente del mandato CL generado puede utilizarse para:

- Mover configuraciones de sistema a sistema
- Mantener configuraciones in situ
- Salvar configuraciones (sin utilizar SAVSYS)

2.6.3 Recuperación del Estado de Configuración

Utilizando el mandato Recuperar Estado de Configuración (RTVCFGSTS), puede ofrecer a las aplicaciones la posibilidad de recuperar el estado de configuración de tres objetos de configuración: línea, controlador y dispositivo. El mandato RTVCFGSTS puede utilizarse en un procedimiento CL para comprobar el estado de una descripción de configuración.

2.6.4 Recuperación de Atributos de Red

Utilizando el mandato Recuperar Atributos de Red (RTVNETA), puede recuperar los atributos de red del sistema. Estos atributos pueden cambiarse utilizando el mandato Cambiar Atributos de Red (CHGNETA) y visualizarse con el mandato Visualizar Atributos de Red (DSPNETA). Consulte el manual *Gestión de Trabajos* para obtener más información referente a atributos de red.

Subtemas
2.6.4.1 Ejemplo de RTVNETA

2.6.4.1 Ejemplo de RTVNETA

En el siguiente ejemplo, se recuperan la cola de salida de la red por omisión y la biblioteca que la contiene, se cambian por QGPL/QPRINT, y después vuelven a cambiarse al valor anterior.

```
  PGM
  DCL  VAR(&OUTQNAME) TYPE(*CHAR) LEN(10)
  DCL  VAR(&OUTQLIB) TYPE(*CHAR) LEN(10)
  RTVNETA  OUTQ(&OUTQNAME) OUTQLIB(&OUTQLIB)
  CHGNETA  OUTQ(QGPL/QPRINT)
  .
  .
  .
  CHGNETA  OUTQ(&OUTQLIB/&OUTQNAME)
  ENDPGM
```

2.6.5 Recuperación de Atributos de Trabajo

Puede recuperar los atributos de trabajo y situar los valores en una variable CL para controlar sus aplicaciones.

Los atributos de trabajo se recuperan utilizando el mandato Recuperar Atributos de Trabajo (RTVJOBA). Con este mandato, puede recuperar todos los atributos del trabajo o cualquier combinación de ellos.

En el siguiente procedimiento CL, un mandato RTVJOBA recupera el nombre del usuario que llamó al procedimiento.

```
          PGM
           /* ORD410C Programa de entrada de pedidos */
          DCL &CLKNAM TYPE(*CHAR) LEN(10)
          DCL &NXTPGM TYPE(*CHAR) LEN(3)
          .
          .
          .
          RTVJOBA USER(&CLKNAM)
  BEGIN:  CALL ORD410S2 PARM(&NXTPGM &CLKNAM)
           /* Solicitud cliente */
          IF (&NXTPGM *EQ 'END') THEN(RETURN)
          .
          .
          .
```

La variable &CLKNAM, en la que se pasa el nombre de usuario, es la primera declarada utilizando un mandato DCL. El mandato RTVJOBA sigue a los mandatos de declaración. Cuando se llama al programa ORD410S2, se le pasan dos variables, &NXTPGM y &CLKNAM. &NXTPGM se pasa como blancos, pero podría cambiarse por ORD410S2.

Subtemas
2.6.5.1 Ejemplo de RTVJOBA

2.6.5.1 Ejemplo de RTVJOBA

Suponga que, en el siguiente procedimiento CL, un trabajo interactivo somete un programa por lotes incluyendo el procedimiento CL. El mandato Recuperar Atributos de Trabajo (RTVJOBA) recupera el nombre de la cola de mensajes a la que se envía el mensaje de conclusión del trabajo y utiliza dicha cola para comunicarse con el usuario que envió el trabajo.

```
  PGM
  DCL &MSGQ *CHAR 10
  DCL &MSGQLIB *CHAR 10
  DCL &MSGKEY *CHAR 4
  DCL &REPLY *CHAR 1
  DCL &ACCTNO *CHAR 6
  .
  .
  .
  RTVJOBA SBMMSGQ(&MSGQ) SBMMSGQLIB(&MSGQLIB)
  IF (&MSGQ *EQ '*NONE') THEN(DO)
     CHGVAR &MSGQ 'QSYSOPR'
     CHGVAR &MSGQLIB 'QSYS'
  ENDDO
  .
  .
  .
  IF (. . . ) THEN(DO)
   SNDMSG:SNDPGMMSG MSG('Número cuenta ' *CAT &ACCTNO *CAT 'no +
                    es válido. ¿Desea cancelar la actualización +
                    (Y o N)?') TOMSGQ(&MSGQLIB/&MSGQ) MSGTYPE(*INQ) +
                    KEYVAR(&MSGKEY)
   RCVMSG MSGQ(*PGMQ) MSGTYPE(*RPY) MSGKEY(&MSGKEY) +
            MSG(&REPLY) WAIT(*MAX)
   IF (&REPLY *EQ 'Y') THEN(RETURN)
   ELSE IF (&REPLY *NE 'N') THEN(GOTO SNDMSG)
  ENDDO
  .
  .
  .
```

Se declaran dos variables, &MSGQ y &MSGQLIB, para recibir el nombre y la biblioteca de la cola de mensajes que se utilizará. El mandato RTVJOBA se utiliza para recuperar el nombre de la cola de mensajes y el nombre de biblioteca. Puesto que es posible que no haya una cola de mensajes especificada para el trabajo, se compara el nombre de cola de mensajes con el valor *NONE. Si son iguales, no se especifica ninguna cola de mensaje y las variables se modifican para que se utilice la cola de mensajes QSYSOPR en la biblioteca QSYS. Posteriormente en el procedimiento, cuando se detecta una condición de error, se envía un mensaje de consulta a la cola de mensajes especificada y se recibe y procesa la respuesta. Otros posibles usos del mandato RTVJOBA son:

- Recuperar uno o más atributos de trabajo (tales como la cola de salida y la lista de bibliotecas) de forma a que puedan cambiarse temporalmente y restaurarse posteriormente a sus valores originales.
- Recuperar uno o más atributos de trabajo para utilizarlos en el mandato SBMJOB, de modo que el trabajo sometido tenga los mismos atributos que el trabajo que somete.

2.6.6 Recuperación de Descripciones de Objeto

También puede utilizar el mandato Recuperar Descripción de Objeto (RTVOBJD) para devolver las descripciones de un objeto determinado a un procedimiento CL. Para devolver las descripciones se utilizan variables. Puede utilizar estas descripciones para ayudarle a detectar objetos no utilizados. Para obtener más información sobre la recuperación de descripciones de objetos, véase el apartado "Recuperación de Descripciones de Objeto" en el tema 4.8.

Puede utilizar la interfaz de programación de aplicaciones (API) Recuperar Descripción de Objeto (QUSROBJD) para devolver la descripción de un objeto específico a un procedimiento. Se utiliza una variable para devolver las descripciones. Para mas información, consulte el manual System API Reference.

2.6.7 Recuperación de Atributos de Perfil de Usuario

Utilizando el mandato Recuperar Atributos de Perfil de Usuario (RTVUSRPRF), puede recuperar los atributos de un perfil de usuario (excepto la contraseña) y colocar los valores en variables CL para controlar las aplicaciones. En este mandato, puede especificar un nombre de perfil de usuario de 10 caracteres o *CURRENT.

También puede supervisar los mensajes de escape después de ejecutar el mandato RTVUSRPRF. Consulte el manual CL Reference para ver una lista de los mensajes que pueden enviarse.

Subtemas
2.6.7.1 Ejemplo de RTVUSRPRF

2.6.7.1 Ejemplo de RTVUSRPRF

En el siguiente procedimiento CL, el mandato RTVUSRPRF recupera el nombre del usuario que llamó al procedimiento y el nombre de una cola de mensajes a la que enviar mensajes para dicho usuario:

```
  DCL  &USR  *CHAR  10
  DCL  &USRMSGQ  *CHAR  10
  DCL  &USRMSGQLIB  *CHAR  10
  .
  .
  .
  RTVUSRPRF  USRPRF(*CURRENT)  RTNUSRPRF(&USR)  +
             MGSQ(&USRMSGQ)  MSGQLIB(&USRMSGQLIB)
```

Se devuelve la información siguiente al procedimiento:

- &USR contiene el nombre de perfil de usuario del usuario que ha llamado al programa.
- &USRMSGQ contiene el nombre de la cola de mensajes especificada en el perfil de usuario.
- &USRMSGQLIB contiene el nombre de la biblioteca que tiene la cola de mensajes asociada con el perfil de usuario.

2.6.8 Recuperación de Información de Descripción de Miembro

Mediante el mandato Recuperar Descripción de Miembro (RTVMBRD), puede recuperar información sobre un miembro de un archivo de base de datos para utilizarlas en sus aplicaciones.

Subtemas
2.6.8.1 Ejemplo de RTVMBRD

2.6.8.1 Ejemplo de RTVMBRD

En el siguiente procedimiento CL, un mandato RTVMBRD recupera la descripción de un miembro específico. Suponga que existe un archivo de base de datos denominado MFILE en la biblioteca actual (MYLIB) que contiene tres miembros (AMEMBER, BMEMBER y CMEMBER).

```
  DCL  &LIB      TYPE(*CHAR) LEN(10)
  DCL  &MBR      TYPE(*CHAR) LEN(10)
  DCL  &SYS      TYPE(*CHAR) LEN(4)
  DCL  &MTYPE    TYPE(*CHAR) LEN(5)
  DCL  &CRTDATE  TYPE(*CHAR) LEN(13)
  DCL  &CHGDATE  TYPE(*CHAR) LEN(13)
  DCL  &TEXT     TYPE(*CHAR) LEN(50)
  DCL  &NBRRCD   TYPE(*DEC)  LEN(10 0)
  DCL  &SIZE     TYPE(*DEC)  LEN(10 0)
  DCL  &USEDATE  TYPE(*CHAR) LEN(13)
  DCL  &USECNT   TYPE(*DEC)  LEN(5 0)
  DCL  &RESET    TYPE(*CHAR) LEN(13)
  .
  .
  .
  RTVMBRD    FILE(*CURLIB/MYFILE) MBR(AMEMBER *NEXT) +
             RTNLIB(&LIB) RTNSYSTEM(&SYS) RTNMBR(&MBR) +
             FILEATR(&MTYPE) CRTDATE(&CRTDATE) TEXT(&TEXT) +
             NBRCURRCD(&NBRRCD) DTASPCSIZ(&SIZE) USEDATE(&USEDATE) +
             USECOUNT(&USECNT) RESETDATE(&RESET)
```

Se devuelve la información siguiente al procedimiento:

- El nombre de biblioteca actual (MYLIB) se coloca en la variable CL llamada &LIB.
- El sistema en el que se encontró MYFILE se coloca en la variable CL &SYS. (*LCL significa que el archivo se encontró en el sistema local y *RMT significa que el archivo se encontró en un sistema remoto.)
- El nombre del miembro (BMEMBER), puesto que BMEMBER es el miembro que hay inmediatamente después de AMEMBER en una lista de miembros clasificados por el nombre (*NEXT), se coloca en la variable CL denominada &MBR.
- El atributo de archivo de MYFILE se coloca en la variable CL denominada &MTYPE. (*DATA significa que el miembro es un miembro de datos y *SRC significa que el archivo es un miembro fuente.)
- La fecha de creación de BMEMBER se coloca en la variable CL denominada &CRTDATE.
- El texto utilizado para describir BMEMBER se coloca en la variable CL denominada &TEXT.
- El número actual de registros en BMEMBER se coloca en la variable CL denominada &NBRRCD.
- El tamaño del espacio de datos de BMEMBER (en bytes) se coloca en la variable CL denominada &SIZE.
- La fecha en que BMEMBER se utilizó por última vez se coloca en la variable denominada &USEDATE.
- El número de días que se ha utilizado BMEMBER se coloca en la variable CL denominada &USECNT. La fecha de inicio de esta cuenta es el valor colocado en la variable CL &RESET.

En QUSRTOOL puede encontrar más ejemplos sobre la utilización de mandatos Recuperar (RTVxxx).

2.7 Trabajar con Procedimientos CL

Un procedimiento fuente CL debe compilarse en un módulo y enlazarse a un programa antes de ejecutarlo.

Para crear un programa CL en un solo paso, utilice el mandato CRTBNDCL y cree un programa enlazado con un módulo.

También puede crear un módulo con el mandato CRTCLMOD. El módulo debe enlazarse a un programa o programa de servicio utilizando el mandato Crear Programa (CRTPGM) o Crear Programa Servicio (CRTSRVPGM).

El siguiente mandato CRTCLMOD crea el módulo ORD040C y lo coloca en la biblioteca DSTPRODLB:

```
  CRTCLMOD     MODULE(DSTPRODLB/ORD040C) SRCFILE(QCLSRC)
               TEXT('Programa menú dept general pedidos')
```

Los mandatos fuente para ORD040C se hallan en el archivo fuente QCLSRC y el nombre de miembro fuente es 0RD040C. Por omisión, se crea un listado del compilador.

En el mandato CRTBNDCL, se pueden especificar opciones de listado y si el programa debe ejecutarse bajo el perfil de usuario del propietario del programa.

Un programa se puede ejecutar utilizando tanto el perfil del propietario como el perfil del usuario.

Los procedimientos y programas CL se crean utilizando opciones del menú Gestor para el Desarrollo de Programas (PDM) o el menú Programador de modo que no debe entrarse directamente CRTCLMOD o CRTBNDCL

Subtemas

2.7.1 Anotación de Mandatos de Procedimiento CL

Puede especificar que la mayoría de mandatos CL que se ejecutan en un procedimiento CL se graben (anoten) en las anotaciones de trabajo especificando uno de los siguientes valores en el parámetro LOG del mandato CRTCLMOD o CRTBNDCL cuando se compila el procedimiento:

*JOB Este valor por omisión indica que la anotación va a producirse cuando la opción de anotaciones del trabajo esté activada. La opción se establece inicialmente para que no haya anotaciones, pero puede cambiarse mediante el parámetro LOGCLPGM en el mandato CHGJOB. Por tanto, si crea el módulo o programa con este valor, puede alterar la opción de anotar para cada trabajo o varias veces dentro de un trabajo.

*YES Este valor indica que la anotación se producirá cada vez que se ejecute el procedimiento CL. No puede cambiarlo mediante el mandato CHGJOB.

*NO Este valor indica que no se producirá ninguna anotación. No puede cambiarlo mediante el mandato CHGJOB.

Debido a que estos valores forman parte del mandato CRTCLMOD o CRTBNDCL, debe volver a compilar el módulo o programa para cambiarlos.

Al especificar la anotación, debe utilizar con precaución el mandato Eliminar Mensaje (RMVMSG) a fin de no eliminar ningún mandato anotado de las anotaciones de trabajo. Si especifica CLEAR(*ALL) en el mandato RMVMSG, en las anotaciones de trabajo no aparecerán los mandatos anotados antes de la ejecución de dicho mandato. Esto sólo afecta al procedimiento CL que contiene el mandato RMVMSG y no afecta a ningún mandato anotado para los niveles de recurrencia anteriores o posteriores.

No todos los mandatos se anotan en las anotaciones de trabajo. A continuación figura la lista de los mandatos que no se anotan:

CHGVAR	DO	GOTO
DCL	ELSE	IF
DCLF	ENDDO	MONMSG
PGM	ENDPGM	CALLPRC

Si la opción de anotación está activada, los mensajes de anotación se envían a la cola de mensajes del procedimiento CL. Si el procedimiento CL se está ejecutando interactivamente, y el nivel de mensaje del parámetro LOG del trabajo está establecido en 4, pulse F10 (Visualizar mensajes de detalle) para ver las anotaciones de todos los mandatos. Puede imprimir la anotación si el nivel de mensajes es 4 y se especifica *PRINT al desconectarse.

La anotación incluye la hora, los nombres de programa y procedimiento, los textos de los mensajes y los nombres de mandato. Los nombres de mandatos se califican tal como lo están en la sentencia fuente original. Los parámetros de los mandatos también se anotan; si la información de un parámetro es una variable CL, el contenido de la variable se imprime (excepto el parámetro RTNVAL).

La anotación de los mandatos afecta al rendimiento.

2.7.2 Listas de Compilador de Módulo CL

Cuando crea un módulo CL, puede crear varios tipos de listados utilizando los parámetros OPTION y OUTPUT en el mandato CRTCLMOD.

Los valores del parámetro OPTION y sus significados son:

- *GEN o *NOGEN

  Si se va a crear un módulo (el valor por omisión es *GEN).
- *XREF o *NOXREF

  En caso de que vaya a producir un listado de referencias cruzadas para variables y referencias de datos en la entrada fuente (*XREF es el valor por omisión).

Los valores del parámetro OUTPUT y sus significados son:

- *PRINT - imprimir listado
- *NONE - ningún listado de compilador

El listado creado especificando el parámetro OUTPUT se llama *listado de compilador*. A continuación se muestra un ejemplo de un listado de compilador. Los números situados en un recuadro hacen referencia a las descripciones que siguen al listado.

```
      1

5763SS1 V3R1M0  940909                 Lenguaje de Control              MYLIB/DUMPERR         05/06/94
Programa  . . . . . . . . . . . . . . . . . :   DUMPERR
  Biblioteca  . . . . . . . . . . . . . . . :     MYLIB
Archivo fuente  . . . . . . . . . . . . . . :   QCLSRC
  Biblioteca  . . . . . . . . . . . . . . . :     MYLIB
Nombre miembro fuente . . . . . . . . . . . :   DUMPERR   05/06/94 10:42:26  4
Opciones impresión fuente . . . . . . . . . :   *XREF  *NOSECLVL  *NOEVENTF
Perfil de usuario . . . . . . . . . . . . . :   *USER
Anotaciones de programa . . . . . . . . . . :   *JOB
Tomar grupo activación por omisión  . . . . . :   *YES
Sustituir programa  . . . . . . . . . . . . :   *YES
Release destino . . . . . . . . . . . . . . :   V3R1M0
Autorización  . . . . . . . . . . . . . . . :   *LIBCRTAUT
Secuencia ordenación  . . . . . . . . . . . :   *HEX
Identificador lenguaje. . . . . . . . . . . :   *JOBRUN
Texto . . . . . . . . . . . . . . . . . . . :   Programa de prueba
Optimización  . . . . . . . . . . . . . . . :   *NONE
Vista depuración  . . . . . . . . . . . . . :   *STMT
Compilador. . . . . . . . . . . . . . . . . :   Compilador Lenguaje Control AS/400 IBM  5
    6                       Fuente Lenguaje Control
SEQNBR  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+.
   100- PGM
   200- DCL &ABC *CHAR 10 VALUE('THIS')
   300- DCL &XYZ *CHAR 10 VALUE('THAT')  7
   400- DCL &MNO *CHAR 10 VALUE('OTHER')
   500- CRTLIB LB(LARRY)
* CPD0043 30  Palabra clave LB no válida para este mandato.  9
   600- DLTLIB LIB(MOE
* CPD0013 30  Falta un paréntesis.
   700- MONMSG CPF0000 EXEC(GOTO ERR)
   800- ERROR:
   900- CHGVAR &ABC 'ONE'
  1000- CHGVAR &XYZ 'TWO'
  1100- CHGVAR &MNO 'THREE'
  1200- DMPCLPGM
  1300- ENDPGM
                            * * * * *   F I N   D E   F U E N T E   * * * * *
5763SS1 V3R1M0  940909                 Lenguaje Control                 MYLIB/DUMPERR         05/06/94
                                     Referencias Cruzadas
Variables Declaradas
Nombre           Definido    Tipo            Longitud    Referencias
&ABC                200      *CHAR             10           900
&MNO                400      *CHAR             10          1100
10
&XYZ                300      *CHAR             10          1000
Etiquetas Definidas
Etiqueta         Definido    Referencias  11
ERR              ******         700
* CPD0715 30  Etiq. 'ERR       ' no existe.
ERROR               800
                * * * * *   F I N   D E   R E F E R E N C I A S   C R U Z A D A S   * * * *
5763SS1 V3R1M0  940909               Lenguaje Control                   MYLIB/DUMPERR         05/06/94
                                     Resumen Mensajes
              Gravedad
```

```
   Total         0-9  10-19  20-29  30-39  40-49  50-59  60-69  70-79  80-89  90-99   12
       3           0      0      0      3      0      0      0      0      0      0
   Programa DUMPERR no creado en biblioteca MYLIB. Severidad error máx. 30.  13
                              * * * * *   F I N   D E   R E S U M E N   M E N S A J E S   * * * *
                                 * * * * *   F I N   D E   C O M P I L A C I O N   * * * * *
```

Título:

1 Número de programa, release, nivel de modificación y fecha del OS/400.

2 Fecha y hora de la ejecución del compilador.

3 Número de página del listado.

Prólogo:

4 Valores del parámetro especificados (o valores por omisión si no se especificaron) en el mandato CRTCLMOD. Si el fuente no está en un archivo de base de datos, se omiten el nombre del miembro, la fecha y la hora.

5 Nombre del compilador.

Fuente:

6 Números de secuencia de las líneas (registros) en el fuente. Un guión a continuación de un número de secuencia indica que una sentencia fuente empieza en dicho número de secuencia. La ausencia de guión indica que una sentencia es la continuación de la sentencia anterior.

Los comentarios entre sentencias fuente se manejan como cualquier otra sentencia fuente y tienen números de secuencia.

Consulte el manual *DB2/400 Programación de la base de datos* para obtener información sobre cómo se asignan los números de secuencia.

7 Sentencias fuente.

8 Última fecha en que se cambió o añadió la sentencia fuente. Si el fuente no está en un archivo de base de datos o las fechas han sido restauradas mediante RGZPFM, se omite la fecha.

9 Si se halla un error durante la compilación y puede asignarse a una sentencia fuente específica, se imprime el mensaje de error inmediatamente después de la sentencia fuente. Un asterisco (*) indica que la línea contiene un mensaje de error. La línea contiene el identificador del mensaje, la gravedad y el texto del mensaje.

Para más información sobre errores de compilación, vea el apartado "Errores Detectados Durante la Compilación" en el tema 2.7.3.

Referencias cruzadas:

10 La tabla de variables simbólicas es un listado de referencias cruzadas de las variables declaradas válidas en el programa. En la tabla figuran la variable, el número de secuencia de la sentencia donde se declara la variable, los atributos de la variable y los números de secuencia de las sentencias que hacen referencia a la variable.

11 La tabla de etiquetas es un listado de referencias cruzadas de las etiquetas definidas válidas en el programa. En la tabla figuran la etiqueta, el número de secuencia de la sentencia donde se define la etiqueta y los números de secuencia de las sentencias que hacen referencia a la etiqueta.

Mensajes:

Esta sección no está incluida en el listado ejemplo, porque no se emitió ningún mensaje de error general para el módulo ejemplo. Si hubieran mensajes de error generales para este módulo, esta sección contendría, para cada mensaje, el identificador de mensaje, la gravedad y el mensaje.

Resumen de mensajes:

12 Resumen del número de mensajes emitidos durante la compilación. Se da el número total junto con los totales por gravedad.

13 Se imprime un mensaje de terminación a continuación del resumen de mensajes.

Las secciones del título, prólogo, fuente y resumen de mensajes se imprimen siempre cuando se utiliza la opción *SOURCE. La sección de referencias cruzadas se imprimirá si se especifica la opción *XREF. La sección de mensajes se imprime solamente si se encuentran errores generales.

El compilador CL produce, a partir de los mandatos CL de un miembro fuente CL, un código de sistema intermedio necesario para que se ejecute la función solicitada cuando se ejecuta el programa.

2.7.3 Errores Detectados Durante la Compilación

En el listado de compilador de un módulo, una condición de error que se refiere directamente a un mandato específico se lista después de este mandato. Consulte el apartado "Listas de Compilador de Módulo CL" en el tema 2.7.2 para ver un ejemplo de estos mensajes incorporados. Los mensajes que no se refieren a un mandato específico pero que son más generales se listan en una sección de mensajes de la lista, no incorporados con sentencias fuente.

Los tipos de errores que se detectan en el momento de la compilación incluyen errores de sintaxis, referencias a variables y a etiquetas no definidas y sentencias omitidas. Los siguientes tipos de errores detienen la creación del procedimiento (se ignoran los códigos de gravedad).

- Errores de valores
- Errores de sintaxis
- Errores relacionados con las dependencias entre parámetros en un mandato
- Errores detectados durante la comprobación de validez.

Aunque se encuentre un error que impida la creación del procedimiento, el compilador continúa comprobando si hay errores en el fuente. Ello le permite ver y corregir tantos errores como sea posible antes de intentar crear el módulo o programa de nuevo.

2.7.4 Obtención de un Vuelco de Procedimiento

Puede obtener un vuelco de procedimiento CL durante el proceso de un procedimiento. El vuelco de procedimiento CL consta de una lista de todos los mensajes en la cola de mensajes del procedimiento, y los valores de todas las variables declaradas en el mismo. Esta información puede ser útil para determinar la causa de un problema que afecta el proceso de un procedimiento.

Para obtener un vuelco de un procedimiento CL, efectúe una de las siguientes opciones:

- Ejecute el mandato Volcar Programa CL (DMPCLPGM). Este mandato puede utilizarse solamente en un procedimiento CL, y no finaliza el procedimiento CL.
- Entre D en respuesta al mensaje de consulta CPA0702. Este mensaje se envía cuando un procedimiento CL recibe un mensaje de escape no supervisado. Si el programa se ejecuta en un trabajo interactivo, el mensaje se envía a la cola externa de mensajes del trabajo. Si el programa se ejecuta como un trabajo por lotes, el mensaje se envía a la cola de mensajes del operador del sistema, QSYSOPR.
- Especifique INQMSGRPY(*SYSRPYL) para el trabajo. Consulte el manual *Gestión de Trabajos* para ver una descripción de este atributo de trabajo. La lista de respuestas suministrada por IBM especifica una respuesta de D para el mensaje CPA0702 o CPA0701, que harán que se imprima un vuelco cuando se envía el mensaje de consulta.
- Cambie la respuesta por omisión del mensaje CPA0702 de C (cancelar programa) a D (volcar procedimiento). Se imprimirá un vuelco de procedimiento siempre que se produzca una comprobación de función en un procedimiento CL. Para cambiar el valor por omisión, entre el mandato siguiente:

```
CHGMSGD MSGID(CPA0702) MSGF(QCPFMSG) DFT(D)
```

Nota: El responsable de seguridad u otro usuario con la autorización de actualización para el archivo de mensajes QCPFMSG debe entrar el mandato CHGMSGD.

El cambio del valor por omisión del mensaje produce la impresión de un vuelco bajo una de las condiciones siguientes:

- La cola de mensajes del operador del sistema está en modalidad por omisión y el mensaje se envía desde un trabajo por lotes.
- El usuario de la estación de pantalla pulsa la tecla Intro sin teclear una respuesta, haciendo que se utilice el mensaje por omisión.
- Se especifica INQMSGRPY(*DFT) para el trabajo.

```
          1
5763SS1 V3R1M0 940909                              Vuelco Programa CL                24/05/94  1

Nombre trabajo . . . . . :  DSP04 3   Nombre usuario  . . . :   SMITH  3   Número trabajo. . . . . : 013
Nombre programa. . . . . :  DUMP 4    Biblioteca  . . . . . :   MYLIB 4    Sentencia . . . . . : 120
Nombre módulo  . . . . . :  DUMP      Nombre procedimiento. :   DUMP

                                                  Mensajes


           ID  6                        Tipo                            De                          A
Hora       Mensaje            Grav     Mensaje Texto                   Programa        Inst        P
110503     CPC2102            00       COMP    Bibliot LARRY creada.   QLICRLIB        *N          D
110503     CPF2110            40       ESC     Bibl MOE no encontrada. QLICLLIB        *N          D

                                                  Variables  7

Variable          Tipo        Longitud          Valor                          Valor en Hexadecimal
                                                 *...+....1....+....2....+     * . . . + . . . . 1 . .
&ABC              *CHAR               10        'UNO       '                   D6D5C540404040404040
&XYZ              *CHAR               10        'DOS       '                   E3E6D640404040404040

                                  * * * * *  F I N   D E   V U E L C O  * * * *
```

1 Número de programa, release, nivel de modificación y fecha del OS/400.

2 Fecha y hora en que se imprimió el vuelco.

3 Nombre totalmente calificado del trabajo en el que se ejecutaba el procedimiento.

4 Nombre y biblioteca del programa.

5 Número de la sentencia en ejecución cuando se efectuó el vuelco. Si se trata de un mandato jerarquizado, el número de sentencia es el del mandato exterior.

6 Cada mensaje de la cola de mensajes de llamada, incluyendo la hora en que se envió el mensaje, el ID de mensaje, la gravedad, el tipo, el texto, el programa emisor y número de instrucción, y el programa receptor y número de instrucción.

7 Todas las variables declaradas en el procedimiento, incluyendo el nombre, tipo, longitud, valor y valor hexadecimal de la variable.

Si una variable decimal contiene datos decimales que no son válidos, los valores hexadecimales y de tipo carácter se imprimen como variables *CHAR.

Si el valor de la variable no puede localizarse, se imprime *NOT ADRESSABLE. Esto puede suceder si el procedimiento CL se utiliza en un programa de proceso de mandatos para un mandato que tiene un parámetro con TYPE(*NULL) o PASSVAL(*NULL) especificado, o si se especificó RTNVAL(*YES) para el parámetro y no se ha codificado una variable de retorno en el mandato.

Si una variable se declara como TYPE(*LGL), se muestra en el vuelco como *CHAR con una longitud de 1.

2.7.5 Visualización de Atributos de Módulo

Puede utilizar el mandato Visualizar Módulo (DSPMOD) para visualizar los atributos de un módulo. La información visualizada o impresa puede utilizarse para determinar las opciones especificadas el mandato empleado para crear el módulo.

Para obtener más información acerca de estos mandatos, consulte el manual CL Reference.

2.7.6 Visualización de Atributos de Programa

Puede utilizar el mandato Visualizar Programa (DSPPGM) para visualizar los atributos de un programa. La información visualizada o impresa puede utilizarse para determinar las opciones especificadas el mandato empleado para crear el programa.

Para obtener más información acerca de estos mandatos, consulte el manual CL Reference.

2.7.7 Resumen de Código de Retorno

El parámetro código de retorno (RTNCDE) del mandato RTVJOBA es un valor decimal de 5 dígitos sin posiciones decimales (12345. por ejemplo). El valor decimal indica el estado de programas llamados.

La lista siguiente resume los códigos de retorno utilizados por los lenguajes soportados en el sistema AS/400:

- Programas RPG IV

  Los códigos de retorno enviados por el compilador RPG IV son:

  0 Cuando se crea el programa
  2 Cuando no se crea el programa

  Los códigos de retorno enviados el ejecutar programas RPG IV son:

  0 Cuando se arranca un programa, o mediante la operación CALL antes de llamar a un programa
  1 Cuando un programa finaliza con el indicador LR activado
  2 Cuando un programa finaliza con un error (respuesta de C, D, F o S a un mensaje de consulta)
  3 Cuando un programa finaliza debido a un indicador de parada (H1-H9)

  Los códigos de retorno RPG IV sólo se prueban después de una llamada (CALL):

  - 0 ó 1 indican inexistencia de errores
  - 3 otorga a RPG IV un código de estado de 231
  - Cualquier otro valor proporciona un código de estado RPG IV de 202 (llamada finalizada con error)

  El usuario no puede probar directamente el código de retorno en el programa RPG IV.

- Programas ILE COBOL/400

  Los códigos de retorno enviados al ejecutar programas COBOL/400 son:

  0 Mediante cada sentencia CALL antes de llamar a un programa
  2 Cuando un programa recibe una comprobación de función (CPF9999) o cuando el manejador genérico de excepción de E/S toma el control y no hay un procedimiento USE aplicable.

  Los programas COBOL/400 no pueden recuperar estos códigos de retorno. Un valor 2 de código de retorno envía un mensaje CBE9001 y ejecuta un mandato Reclamar Recursos (RCLRSC) con la opción *ABNORMAL.

- Programas BASIC

  Los programas BASIC compilados o interpretados pueden establecer el código de retorno a cualquier valor comprendido entre -32768 y 32767 codificando una expresión en las sentencias END o STOP. Si no se codifica ninguna expresión, los códigos de retorno se establecen en:

  0 Para terminación normal
  1 Para programas que finalizan a causa de un error

  Los valores de los códigos de retorno BASIC pueden recuperarse utilizando la función intrínseca CODE.

- Programas PL/I

  Los códigos de retorno enviados por los programas PL/I son:

  0 Para terminación normal, establecidos al principio de la unidad de ejecución
  2 Cuando los establece una sentencia STOP o una llamada a PLIDUMP con la opción S
  3 Cuando se activa una condición de ERROR
  4 Cuando PL/I detecta un error que no permitió activar la condición de ERROR

  Si se halla cualquier otro valor, se establece mediante la función incorporada PLIRETC o mediante un procedimiento al que se llama. PLIRETC pasa un código de retorno desde el programa PL/I al programa que lo llamó, cambiando el valor del código de retorno actual.

  Los programas PL/I pueden recuperar el código de retorno utilizando la función incorporada PLIRETV.

- Programas Pascal

El compilador de Pascal establece los códigos de retorno siguientes:

0 Para una compilación satisfactoria
2 Cuando no se produce el programa objeto

Pascal no establece explícitamente el código de retorno en tiempo de ejecución. El usuario puede utilizar la rutina de manejo de excepción ONERROR para supervisar las excepciones concretas y, a continuación, establecer el código de retorno utilizando el procedimiento RETCODE.

Los códigos de retorno Pascal pueden recuperarse utilizando el parámetro RETVALUE del procedimiento SYSTEM de Pascal. El código de retorno se establece a 0 antes de la llamada a SYSTEM y contendrá el código de retorno actualizado una vez devuelto.

- Programa CL

  El valor actual del código de retorno para los programas compilados CL puede recuperarse utilizando el parámetro RTNCDE en el mandato RTVJOBA.

- Programas C/400*

  El valor actual del entero de código de retorno devuelto por la última sentencia de retorno C/400 de un programa C/400.

3.0 Capítulo 3. Control del Flujo y Comunicación entre Programas y Procedimientos

Puede utilizar los mandatos CALL, CALLPRC, y RETURN para que los programas y procedimientos se intercambien el control. Cada mandato tiene unas características distintas. La información se puede pasar como parámetros a programas y procedimientos a los que se llama cuando se pasa el control.

Debe prestar especial atención a los programas creados con USRPRF(*OWNER) que ejecutan los mandatos CALL o CALLPRC. Las características de seguridad de estos mandatos son diferentes cuando se procesan en los programas que se ejecutan bajo el perfil de usuario del propietario. Consulte el manual Security - Reference para mas información acerca de los perfiles de usuario.

Este capítulo incluye Interfaces de Programación de Uso General (GUPI), que proporciona IBM para su uso en programas elaborados por el cliente.

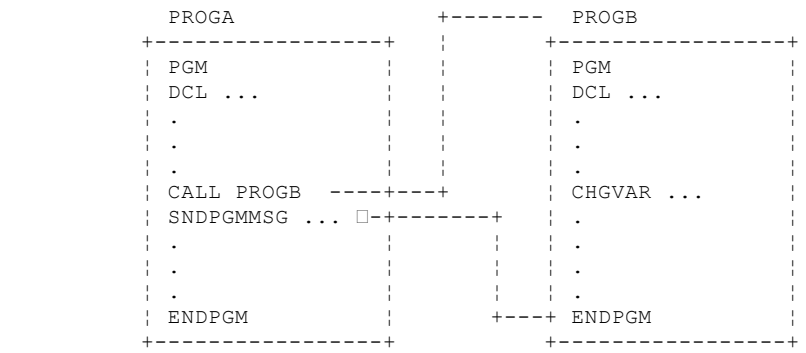
Subtemas

3.1 Mandato CALL

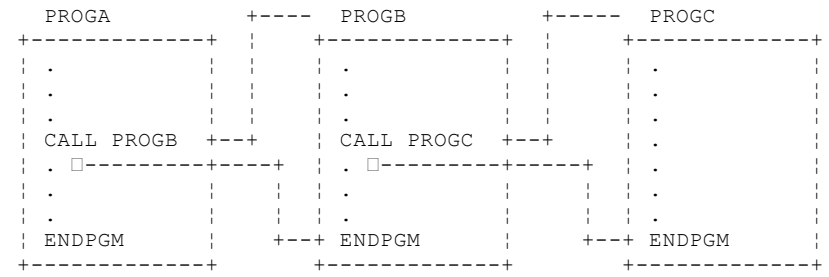
El mandato CALL llama a un programa mencionado en el mandato y le pasa el control. El mandato CALL tiene este formato:

```
  CALL PGM (nombre-biblioteca/nombre-programa) PARM(valores-parámetro)
```

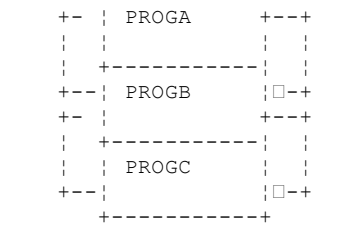
El nombre de programa o nombre de biblioteca puede ser una variable. Si el programa al que se llama se encuentra en un biblioteca que no figura en la lista de bibliotecas, debe especificar el nombre calificado del programa en el parámetro PGM. El parámetro PARM se trata en el apartado "Paso de Parámetros entre Programas y Procedimientos" en el tema 3.4. Cuando finaliza la ejecución del programa al que se llama, el control vuelve al siguiente mandato del programa de llamada.



La secuencia de mandatos CALL en un conjunto de programas que se llaman unos a otros se denomina pila de programas. Por ejemplo, en esta serie:



la pila de llamadas es:



Cuando finaliza el proceso de PROGC, el control vuelve a PROGB, exactamente al mandato situado después de la llamada a PROGC. De este modo, el control se devuelve hacia arriba en la pila de llamadas. Esto sucede tanto si PROGC finaliza con un mandato RETURN o ENDPGM como si no.

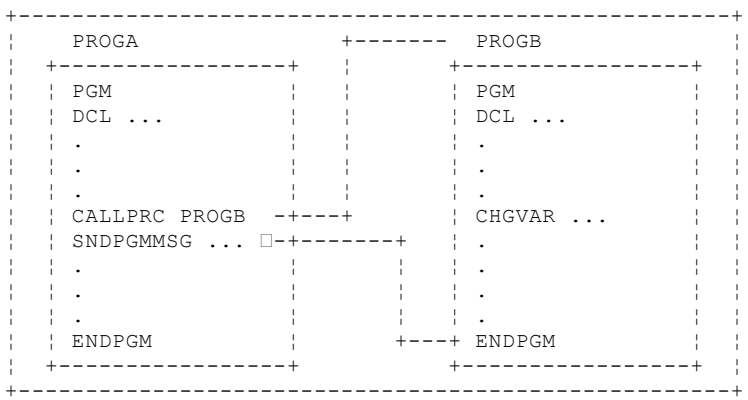
Un programa CL puede llamarse a sí mismo.

3.2 Mandato CALLPRC

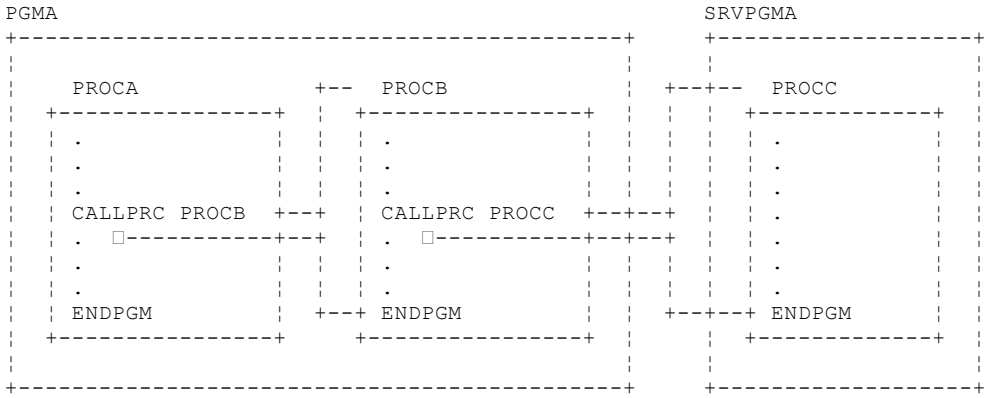
El mandato CALLPRC llama a un procedimiento mencionado en el mandato y le pasa el control. El mandato CALLPRC tiene el siguiente formato:

```
CALLPRC Procedimiento(nombre-proced) PARM(val-parám) RTNVAL(var-valor-retorno)
```

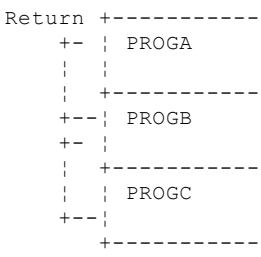
El procedimiento puede no ser una variable. El parámetro PARM se trata en el apartado "Paso de Parámetros entre Programas y Procedimientos" en el tema 3.4. Cuando el procedimiento llamado finaliza la ejecución, el control vuelve al siguiente mandato del procedimiento de llamada.



La secuencia de mandatos CALLPRC en un conjunto de procedimientos que se llaman unos a otros se denomina pila de llamadas. Por ejemplo, en esta serie:



la pila de llamadas es:



Cuando finaliza el proceso de PROGC, el control vuelve a PROGB, exactamente al mandato situado después de la llamada a PROGC. De este modo, el control se devuelve hacia arriba en la pila de llamadas. Esto sucede tanto si PROGC finaliza con un mandato RETURN o ENDPGM como si no.

Un procedimiento CL puede llamarse a sí mismo.

3.3 Mandato RETURN

El mandato RETURN de un procedimiento CL o programa OPM elimina ese procedimiento o programa OPM de la pila de llamadas.

Si el procedimiento que contiene el mandato RETURN se llamó desde el mandato CALLPRC, el control se devuelve a la siguiente sentencia en secuencia después del mandato CALLPRC del programa que efectúa la llamada.

Si un mandato MONMSG especifica una acción que finaliza con un mandato RETURN, el control vuelve a la siguiente sentencia en secuencia que sigue a la sentencia que llamó al procedimiento o programa que contiene el mandato MONMSG.

El mandato RETURN no tiene parámetros.

Nota: Si tiene un mandato RETURN en un programa inicial, se visualiza la pantalla de entrada de mandatos. Conviene evitar esto por motivos de seguridad.

3.4 Paso de Parámetros entre Programas y Procedimientos

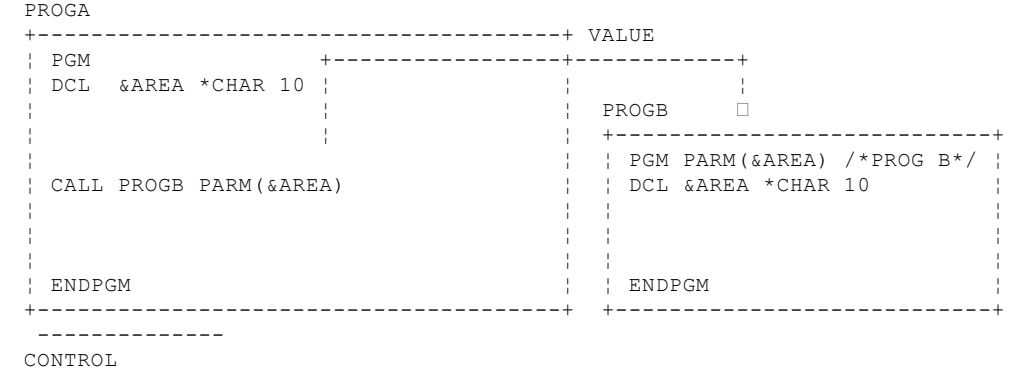
Cuando pasa el control a otro programa o procedimiento, también puede pasar información para realizar modificaciones o para utilizarla en el programa o procedimiento receptor. Para más información, véase el apartado "Utilización del Mandato CALL" en el tema 3.4.1. Puede especificar la información que se va a pasar en el parámetro PARM en un mandato CALL o CALLPRC. Las características y requisitos de estos mandatos son ligeramente distintos.

Por ejemplo, si PROGA contiene el siguiente mandato:

```
  CALL PROGB PARM(&AREA)
```

llama a PROGB y le pasa el valor de &AREA. PROGB debe iniciarse con el mandato PGM, que también debe especificar el parámetro que va a recibir:

```
  PGM PARM(&AREA) /* PROGB */
```



Para el mandato CALL o CALLPRC, debe especificar los parámetros pasados en el parámetro PARM, y debe especificarlos en el parámetro PARM del mandato PGM del programa o procedimiento receptor. Debido a que los parámetros se pasan por posición, no por nombre, la posición del valor pasado en el mandato CALL o CALLPRC debe ser la misma que en el mandato PGM receptor. Por ejemplo, si PROGA contiene el siguiente mandato:

```
  CALL PROGB PARM(&A &B &C ABC)
```

pasa tres variables y una serie de caracteres y, si PROGB empieza de este modo:

```
  PGM PARM(&C &B &A &D) /*PROGB*/
```

entonces el valor de &A en PROGA se utiliza para &C en PROGB, y así sucesivamente; &D en PROGB es **ABC**. El orden de las sentencias DCL en PROGB no es importante. Sólo el orden en que los parámetros están especificados en la sentencia PGM determina qué variables se pasan.

Además de la posición de los parámetros, debe prestarse una especial atención a su longitud y tipo. Los parámetros listados en el programa o procedimiento receptor deben declararse con la misma longitud y tipo que tienen en el programa o procedimiento de llamada. Las constantes decimales se pasan siempre con una longitud de **(15 5)**.

Cuando utiliza el mandato CALLPRC y pasa constantes de serie de caracteres, debe especificar el número exacto de bytes y pasar exactamente ese número. El procedimiento llamado puede utilizar la información del descriptor operativo para determinar el número exacto de bytes pasados. Puede utilizar la API CEEDOD para acceder al descriptor operativo. Consulte la publicación System API Reference para obtener información acerca de API CEEDOD.

Cuando utiliza el mandato CALL, las constantes de serie de caracteres de 32 bytes o menos siempre se pasan con una longitud de 32 bytes. Si la serie tiene más de 32 bytes, debe especificar el número exacto de bytes y pasar exactamente ese número.

A continuación se presenta un ejemplo de un procedimiento o programa que recibe el valor &VAR1:

```
  PGM PARM(&VAR1)  /*PGMA*/
  DCL VAR1 *CHAR LEN(36)
  .
  .
```

```
  .
  ENDPGM
```

El mandato CALL o CALLPRC debe especificar 36 caracteres:

```
  CALLPRC PGMA(ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJ)
```

El ejemplo siguiente especifica las longitudes por omisión:

```
  PGM PARM(&P1 &P2)
  DCL VAR(&P1) TYPE(*CHAR) LEN(32)
  DCL VAR(&P2) TYPE(*DEC) LEN(15 5)
  IF (&P1 *EQ DATA) THEN(CALL MYPROG &P2)
  ENDPGM
```

Para llamar a este programa, podría especificar:

```
  CALL PROG (DATA 136)
```

La serie de caracteres DATA se pasa a &P1; el valor decimal 136 se pasa a &P2.

Si se trabaja con variables definidas localmente, se produce una menor actividad general que si se trabaja con variables pasadas. Por lo tanto, si el procedimiento o programa llamado frecuentemente hace referencia a variables pasadas, puede mejorarse el rendimiento copiando los valores pasados a una variable local y haciendo referencia al valor definido localmente en vez de al valor pasado.

Subtemas

3.4.1 Utilización del Mandato CALL

Cuando un procedimiento CL emite el mandato CALL, cada valor del parámetro que se ha pasado al programa al cual se llama puede ser una constante de serie de caracteres, una constante numérica, una constante lógica o una variable CL. A dicho programa se le pueden pasar un máximo de 40 parámetros. Los valores de los parámetros se pasan según el orden en que aparecen en el mandato CALL, el cual debe coincidir con la lista de parámetros del programa llamado. Los nombres de las variables pasadas no tienen por qué ser los mismos que los de la lista de recepción de parámetros. Los nombres de las variables que reciben los valores del programa al que se llama deben declararse en dicho programa, pero el orden de los mandatos de declaración carece de importancia.

No se asocia ningún almacenamiento en el programa al que se llama con las variables que recibe. En su lugar, cuando se pasa una variable, el almacenamiento para la variable está en el programa en el que se declaró originalmente. Las variables se pasan por dirección. Cuando se pasa una constante, se efectúa una copia de la misma en el programa de llamada y esa copia se pasa al programa al que se llama.

El resultado es que cuando se pasa una variable, el programa llamado puede cambiar el valor de la variable y el cambio se refleja en el programa de llamada. El nuevo valor no tiene que devolverse al programa de llamada para una utilización posterior; ya está allí. Así pues, no se necesita ninguna codificación especial para una variable que va a devolverse al programa de llamada. Cuando se pasa una constante y el programa al que se llama cambia su valor, el programa de llamada no sabe cuál es el valor cambiado. Así pues, si el programa de llamada, llamara de nuevo al mismo programa, volvería a inicializar los valores de las constantes, pero no los de las variables.

Una excepción a lo expuesto anteriormente es el uso del mandato CALL para llamar a un programa C/400. En este caso, los parámetros del mandato CALL (constantes) se convierten en series que terminan en nulos (requisito estándar de ANSIC) y se pasan por valor. Consulte la publicación *IBM SAA C/400 User's Guide, SC09-1347* para obtener mas información.

Si se llamase a un programa CL utilizando un mandato CALL que no se ha compilado (un mandato CALL interactivo o mediante el mandato SBMJOB), los parámetros decimales (*DEC) tendrían que declararse con **LEN(15 5)**, y los parámetros de tipo carácter (*CHAR) deberían declararse con **LEN(32)** o con una longitud menor en el programa receptor.

Un mandato CALL que no está en un programa o procedimiento CL no puede pasar variables como argumentos. Cuando se utiliza el mandato CALL con parámetros de mandato definidos como *CMDSTR, el contenido de las variables especificadas en el parámetro PARMS se convierten a constantes. Algunos ejemplos son los parámetros de mandato (CMD) del mandato someter trabajo (SBMJOB), añadir trabajo (ADDJOBSCDE) o cambiar trabajo (CHGJOBSCDE). Para obtener mas información sobre como se pasan los parámetros al utilizar un mandato CALL interactivo, consulte la descripción del mandato CALL en el manual CL Reference.

Los parámetros pueden pasarse y recibirse del modo siguiente:

- Las constantes de serie de caracteres de 32 bytes o menos se pasan *siempre* con una longitud de 32 bytes (rellenadas a la derecha con espacios en blanco). Si una constante de tipo carácter tiene más de 32 bytes, se pasa toda la longitud de la constante. Si el parámetro se ha definido para contener más de 32 bytes, el mandato CALL debe pasar una constante que contenga exactamente ese número de bytes. Las constantes de más de 32 caracteres ***no*** se rellenan con la longitud esperada por el programa receptor.

  El programa receptor puede recibir menos bytes de los pasados. Por ejemplo, si un programa especifica que han de recibirse 4 caracteres y se pasa **ABCDEF** (rellenado con 26 espacios en blanco), el programa sólo acepta y utiliza **ABCD**.

  Si el programa receptor recibe más bytes de los pasados, el resultado puede ser imprevisible. Los valores numéricos pasados como caracteres deben colocarse entre apóstrofos.

  El sistema contabiliza los caracteres en un mandato SMBJOB. Se empieza a contar a partir del primer carácter que sigue al paréntesis derecho. El nombre del mandato, los nombres de parámetros, los espacios, las longitudes de las variables y los apóstrofos se cuentan como caracteres. Si utiliza variables de caracteres en la serie de mandatos, cada vez que se sustituye el valor se añaden unos apóstrofos que se contabilizarán de igual forma.

- Las constantes decimales se pasan en formato empaquetado y con una longitud de **LEN(15 5)**, lo que significa que el valor tiene 15 dígitos

de longitud, de los cuales 5 son posiciones decimales. Así, si se pasa un parámetro de **12345**, el programa receptor debe declarar el campo decimal con una longitud de **LEN(15 5)**; el parámetro se recibe como **12345,00000**.

Si necesita pasar una constante numérica a un programa y el éste espera un valor con una longitud y precisión diferentes de 15 5, la constante puede codificarse en formato hexadecimal. El mandato CALL siguiente muestra cómo pasar el valor 25,5 a una variable de programa que se declara como LEN(5 2):

```
  CALL PGMA PARM(X'02550F')
```

- Las constantes lógicas se pasan con una longitud de 32 bytes. El valor lógico 0 ó 1 está en el primer byte, y los demás bytes están en blanco. Si se pasa un valor distinto de 0 ó 1 a un programa que espera un valor lógico, los resultados pueden ser imprevisibles.

- Un literal con coma flotante o un valor especial con coma flotante (*NAN, *INF, o *NEGINF) se pasa como un valor de precisión doble que ocupa 8 bytes. Aunque un programa CL no pueda procesar los números con coma flotante, puede recibir un valor con coma flotante en una variable de tipo carácter y pasar dicha variable a un programa HLL que pueda procesar valores con coma flotante.

- Una variable puede pasarse si la llamada se realiza desde un programa o procedimiento CL, en cuyo caso el programa receptor debe declarar el campo para que coincida con la variable definida en el procedimiento o programa que efectúa la llamada. Por ejemplo, si un programa o procedimiento CL define una variable decimal &CHKNUM como **LEN(5 0)**, el programa receptor debe declarar el campo como empaquetado con 5 dígitos en total, sin posiciones decimales. Cuando se ejecuta un mandato CALL en modalidad por lotes mediante el mandato SBMJOB en un programa o procedimiento CL, las variables que se pasan como argumentos se tratan como constantes.

- Si una constante decimal o una variable de programa puede pasarse al programa al que se llama, el parámetro debe definirse como **LEN(15 5)**, y cualquier programa de llamada debe aceptar esta definición. Si el tipo, número, orden y longitud de los parámetros de los programas de llamada y receptor no coinciden (salvo la excepción de longitud anteriormente mencionada para las constantes de tipo carácter), los resultados no pueden predecirse.

- El valor *N no puede utilizarse para especificar un valor nulo porque no puede pasarse un valor nulo a otro programa.

En el ejemplo siguiente, el programa A pasa seis parámetros: una constante lógica, tres variables, una constante de tipo carácter y una constante numérica.

```
  PGM /* PROGRAMA A */
  DCL VAR(&B) TYPE(*CHAR)
  DCL VAR(&C) TYPE(*DEC) LEN(15 5) VALUE(13.529)
  DCL VAR(&D) TYPE(*CHAR) VALUE('1234.56')
  CHGVAR VAR(&B) VALUE(ABCDEF)
  CALL PGM(B) PARM('1' &B &C &D XYZ 2)  /* Observar los blancos entre parms */
  .
  .
  .
  ENDPGM

  PGM PARM(&A &B &C &W &V &U) /* PROGRAMA B */
  DCL VAR(&A) TYPE(*LGL)
  DCL VAR(&B) TYPE(*CHAR) LEN(4)
  DCL VAR(&C) TYPE(*DEC)
       /* Longitud por omisión (15 5) coincidencias DCL LEN en programa A */
  DCL VAR(&W) TYPE(*CHAR)
  DCL VAR(&V) TYPE(*CHAR)
  DCL VAR(&U) TYPE(*DEC)
  .
  .
  .
  ENDPGM
```

Nota: Si el quinto parámetro pasado a PGMB fuese **456** en vez de **XYZ** y quisiera que fuera un dato alfanumérico, este valor debería haberse especificado como '**456**' en el parámetro.

La constante lógica '1' no tiene que declararse en el programa que efectúa la llamada. Se declara como de tipo lógico y se denomina &A en el programa B.

Debido a que no hay una longitud especificada en el mandato DCL para &B, se pasa la longitud por omisión, que es de 32 caracteres. Solo están especificados 6 caracteres de &B (**ABCDEF**). Debido a que &B se ha declarado en el programa B con 4 caracteres, solo se recibe este número de caracteres. Si se cambian en el programa B, también se cambiarán esas 4 posiciones para &B en el programa A para el resto de esta llamada.

El parámetro de longitud (LEN) debe estar especificado para &C en el programa A. Si no se hubiera especificado, la longitud tomaría como valor por omisión la longitud del valor especificado, que sería incompatible con la longitud por omisión esperada en el programa B. &C tiene un valor de **13,52900.**

&W en el programa B (&D en el programa A) se recibe como un carácter, ya que se ha declarado como un carácter. Los apóstrofos no son necesarios para indicar una serie si TYPE es *CHAR. En el programa A, la longitud toma como valor por omisión la longitud del valor, que es 7 (la coma decimal se considera como una posición en una serie de caracteres). El programa B espera una longitud de 32. Se pasan los primeros 7 caracteres, pero no puede saberse cuál será el contenido más allá de la posición 7.

La variable &V es una serie de caracteres **XYZ**, rellenada con blancos por la derecha. La variable &U es un dato numérico, **2.00000.**

Para obtener más información sobre las longitudes por omisión en los mandatos DCL, consulte el mandato DCL en la publicación CL Reference.

3.4.2 Errores Comunes al Llamar a Programas y Procedimientos

En los apartados siguientes se describen los errores detectados con más frecuencia al pasar valores en un mandato CALL o CALLPRC. Algunos de los errores pueden ser muy difíciles de depurar y otros tienen graves consecuencias en las funciones del programa.

Subtemas

3.4.2.1 Errores de Tipo de Datos con la Utilización del Mandato CALL

La longitud total de la serie de mandatos incluye el nombre de los mandatos, los espacios, los nombres de los parámetros, los paréntesis, el contenido de las variables y apóstrofos utilizados.

En la mayor parte de los mandatos, la serie de mandatos inicia el programa de proceso de mandatos tal como se esperaba. Sin embargo, en algunos mandatos es posible que algunas variables no se pasen tal como se esperaba. Para obtener más información sobre las variables, véase el apartado "Trabajar con Variables" en el tema 2.4.

Cuando el mandato CALL se utiliza con el parámetro CMD en el mandato SBMJOB, pueden producirse resultados inesperados. La sintaxis del mandato CALL es la misma cuando se utiliza con el parámetro CMD que cuando se utiliza como directriz del compilador. Cuando se emplea con el parámetro CMD, el mandato CALL se convierte en una serie de mandatos que se ejecutan posteriormente cuando el subsistema por lotes la inicia. Cuando se utiliza por sí solo, el compilador CL genera código para efectuar la llamada.

A menudo surgen problemas con las constantes decimales y las variables de tipo carácter. En los siguientes casos, la serie de mandatos no se crea tal como es necesario:

- Cuando los números decimales se convierten en constantes decimales.

  Cuando se ejecuta la serie de mandatos, la constante decimal se pasa en formato empaquetado con una longitud de LEN(15 5), no se pasa en el formato especificado por la variable CL.
- Cuando una variable de tipo carácter se declara con una longitud superior a los 32 caracteres.

El contenido de la variable de tipo carácter se pasa como se ha descrito anteriormente, generalmente como una constante de tipo carácter entre comillas en la que se han suprimido los blancos de cola. Como resultado, es posible que no se pasen suficientes datos al programa al que se llama.

Los siguientes métodos se pueden utilizar para corregir los errores de construcción de series de mandatos:

- Cree la serie del mandato CALL para someterla mediante la concatenación de las diversas partes del mandato en una sola variable CL. Someta la serie de mandato utilizando el parámetro de petición de datos (RQSDTA) del mandato SBMJOB.

- En el caso de las variables CL de tipo carácter de más de 32 caracteres en las que los blancos de cola son significativos, cree una variable cuya longitud sea de un carácter más de lo necesario y cree una subserie que no sea de caracteres en blanco en la última posición. Ello impide que los espacios en blanco significativos se trunquen. El programa al que se llama debe pasar por alto este carácter adicional, ya que excede la longitud esperada.

- Cree un mandato que iniciará el programa que se va a llamar. Someta el nuevo mandato en lugar de utilizar el mandato CALL. La definición del mandato garantiza que los parámetros se pasan al programa de proceso de mandatos, tal como se esperaba.

3.4.2.2 Errores de Tipo de Datos

Al pasar un valor, el tipo de datos (parámetro TYPE) debe ser el mismo (*CHAR, *DEC o *LGL) en el programa o procedimiento de llamada y en el programa o procedimiento al que se llama. En esta área suelen producirse errores cuando se intenta pasar una constante numérica. Si la constante numérica se escribe entre apóstrofos, pasa como si fuese una serie de caracteres. Sin embargo, si la constante no se escribe entre apóstrofos, pasa como un campo numérico empaquetado con LEN(15 5).

En el ejemplo siguiente, se pasa un valor numérico entrecomillado a un programa que espera un valor decimal. Se produce un error de datos decimales (mensaje de escape MCH1202) cuando se hace referencia a la variable &A en el programa al que se llama (PGMA):

```
  CALL   PGMA PARM('123')  /* PROGRAMA DE LLAMADA */
  PGM    PARM(&A)  /* PGMA */
  DCL    &A *DEC LEN(15 5)  /* LONGITUD POR OMISIÓN */
  .
  .
  .
  IF (&A *GT 0) THEN(...) /* MCH1202 SE PRODUCE AQUÍ */
```

En el ejemplo siguiente, se pasa un valor decimal a un programa definiendo una variable de tipo carácter. Normalmente, este error no ocasiona anomalías en la ejecución, pero, con frecuencia, el resultado es incorrecto.

```
  CALL PGMB PARM(12345678)  /* PROGRAMA DE LLAMADA */

  PGM PARM(&A)               /* PGMB */
  DCL &A *CHAR 8
  .
  .
  .
  ENDPGM
```

La variable &A en PGMB tiene un valor hexadecimal de **0012345678000000F**.

Generalmente, los datos se pueden pasar de una variable lógica (*LGL) a una variable de tipo carácter (*CHAR) y viceversa, sin errores, mientras el valor se exprese como '0' ó '1'.

3.4.2.3 Errores de Longitud Decimal y Precisión

Si un valor decimal se pasa con una longitud decimal y precisión incorrecta (demasiado largo o demasiado corto), se produce un error de datos decimales (MCH1202) cuando se hace referencia a la variable. En los ejemplos siguientes, la constante numérica se pasa como **LEN(15 5)**, pero en el procedimiento o programa llamado se declara como **LEN(5 2)**. Las constantes numéricas siempre se pasan como decimales empaquetados **(15 5)**.

```
  CALL PGMA PARM(123)     /* PROGRAMA DE LLAMADA */

  PGM PARM(&A)                  /* PGMA */
  DCL &A *DEC (5 2)
  .
  .
  .
  IF (&A *GT 0) THEN(...) /* MCH1202 OCURRE AQUÍ */
```

Si una variable decimal se ha declarado con **LEN(5 2)** en el programa o procedimiento de llamada y el valor se ha pasado como variable en vez de como constante, no se producirá ningún error.

Si necesita pasar una constante numérica a un programa o procedimiento y éste espera un valor con una longitud y precisión diferentes de **15 5**, la constante puede codificarse en formato hexadecimal. El mandato CALL siguiente muestra cómo pasar el valor **25.5** a una variable de programa que se declara como **LEN(5 2)**:

```
  CALL PGMA PARM(X'02550F')
```

Si se pasa un valor decimal con la longitud correcta pero con una precisión (número de posiciones decimales) errónea, el programa o procedimiento receptor interpreta el valor incorrectamente. En el ejemplo siguiente, el valor de constante numérica (cuya longitud es (15 5)) pasado al procedimiento se maneja como **25124,00**.

```
  CALL PGMA PARM(25.124)  /* PROGRAMA DE LLAMADA */

  PGM PARM(&A)            /* PGMA */
  DCL &A *DEC (15 2)      /* LEN DEBERÍA SER 15 5*/
  .
  .
  .
  ENDPGM
```

Estos errores se producen cuando se hace referencia a la variable por primera vez, no cuando se pasa o se declara. En el ejemplo siguiente, el programa al que se llama no hace referencia a la variable, sino que solamente coloca un valor (el de la longitud errónea detectada) en la variable devuelta al programa de llamada. El error no se detecta hasta que la variable se devuelve al programa de llamada y se hace referencia a ella por primera vez. Este tipo de error puede ser especialmente difícil de detectar.

```
  PGM        /* PGMA */
  DCL &A *DEC (7 2)
  CALL PGMB PARM(&A)  /* (7 2) PASADO A PGMB */
  IF (&A *NE 0)  THEN(...)  /* *MCH1202 SE PRODUCE AQUÍ */
  .
  .
  .
  ENDPGM

  PGM PARM(&A) /* PGMB */
  DCL &A *DEC (5 2)        /* LONGITUD ERRóNEA */
  .
  .
  .
  CHGVAR &A (&B-&C)      /* VALOR SITUADO en &A */
  RETURN
```

Cuando el control se devuelve al programa PGMA, y se referencia a &A, se produce el error.

3.4.2.4 Errores de Longitud de Caracteres

Si pasa un valor cuya longitud es superior a la de los caracteres declarados de la variable receptora, el procedimiento o programa receptor no puede acceder a la longitud sobrepasada. En el ejemplo siguiente, PGMB cambia la variable que se le pasa por espacios en blanco. Como la variable está declarada con **LEN(5)**, sólo se cambian 5 caracteres por espacios en blanco en PGMB, mientras que los caracteres restantes continúan formando parte del valor cuando se les hace referencia en PGMA.

```
PGM        /* PGMA */
DCL &A *CHAR 10
CHGVAR &A 'ABCDEFGHIJ'
CALL PGMB PARM(&A)           /* PASA A PGMB */
.
.
.
IF (&A *EQ '  ') THEN(...)  /* ESTA PRUEBA FALLA */
ENDPGM

PGM PARM(&A)     /* PGMB */
DCL &A *CHAR 5   /* ERROR DE LEN */
CHGVAR &A '  '   /* SOLO 5 POSICIONES; RESTO NO AFECTADAS */
RETURN
```

Mientras que este tipo de error no produce un mensaje de escape, las variables que se manejan de esta forma pueden funcionar de forma distinta a lo esperado.

Si el valor pasado a un procedimiento o programa es inferior a su longitud declarada en el programa o procedimiento receptor, las consecuencias pueden ser más graves. En este caso, el valor de la variable en el programa o procedimiento llamado consta de sus valores tal como se pasaron originalmente y de todo lo que siga a dicho valor en el almacenamiento, hasta alcanzar la longitud declarada en el programa o procedimiento receptor. El contenido de este almacenamiento adoptado es imprevisible. Si el valor pasado es una variable del programa, podría estar seguido por otras variables o por estructuras internas de control del programa o procedimiento. Si el valor pasado es una constante, podría estar seguido en el almacenamiento por otras constantes pasadas en el mandato CALL o CALLPRC o por estructuras internas de control.

Si el procedimiento o programa receptor cambia el valor, opera en el valor original y en el almacenamiento adoptado. El efecto inmediato de ello podría ser que se cambiaran otras variables o constantes o que se cambiaran las estructuras internas de tal manera que el programa o procedimiento fallase. Los cambios efectuados al almacenamiento adoptado entran en vigor inmediatamente.

En el ejemplo siguiente, se pasan dos constantes de tres caracteres al programa llamado. Las constantes de tipo carácter se pasan con un mínimo
|de 32 caracteres para el mandato CALL. (Normalmente, el valor se pasa
|como 3 caracteres justificados a la izquierda con blancos de cola.) Si el programa receptor declara la variable receptora con una longitud superior a 32 posiciones, las posiciones adicionales utilizan almacenamiento adoptado de valor desconocido. En este ejemplo, supongamos que las dos constantes son adyacentes en el almacenamiento.

```
CALL PGMA ('ABC' 'DEF')  /* PROG QUE PASA VALOR */

PGM  PARM(&A &B) /* PGMA */
DCL  &A *CHAR 50  /* VALOR:ABC+29' '+DEF+15' ' */
DCL  &B *CHAR 10  /* VALOR:DEF+7' ' */
CHGVAR VAR(&A) (' ')  /* TAMBIEN LOS BLANCOS DE &B */
.
.
.
ENDPGM
```

El comportamiento de los valores pasados en las variables es exactamente el mismo.

| En el ejemplo siguiente, se pasan dos constantes de tres caracteres al
|procedimiento llamado. Al mandato CALLPROC sólo se le pasa el número de
|caracteres especificado. Si el programa receptor declara la variable
|receptora con una longitud superior a la de la constante pasada, las
|posiciones adicionales utilizan almacenamiento adoptado de valor
|desconocido.

|En el siguiente ejemplo, se asume que las dos constantes están adyacentes
|en el almacenamiento.

```
|  CALLPRC PRCA ('ABC' 'DEF')  /* PROGRAMA QUE PASA VALOR */
```

```
|  PGM  PARM(&A &B)        /*  *PRCA  */
|  DCL  &A *CHAR 5         /*   VALOR:'ABC' + 'DE' */
|  DCL  &B *CHAR 3         /*   VALOR:'DEF' */
|  CHGVAR &A ' '           /*   También pone a blancos los dos primeros  */
|                          /*   bytes de &B */
|  .
|  .
|  .
|  ENDPGM
```

3.5 Utilización de Colas de Datos para la Comunicación entre Programas y Procedimientos

Las **colas de datos** son un tipo de objeto del sistema que usted puede crear, al que un procedimiento o programa HLL puede enviar datos, y del que otro procedimiento o programa HLL puede recibir datos. El programa receptor puede o bien estar a la espera de datos o puede recibir los datos más tarde.

Las ventajas de utilizar colas de datos son las siguientes:

- La utilización de colas de datos libera a un trabajo de la realización de alguna tarea. Si el trabajo es interactivo, puede proporcionar un mejor tiempo de respuesta y disminuir el tamaño del programa interactivo y su grupo de acceso de proceso (PAG). Esto, a su vez, puede ayudar al rendimiento general del sistema. Por ejemplo, si varios usuarios de estación de trabajo entran una transacción que implica actualizaciones y adiciones a varios archivos, el sistema puede tener un mejor rendimiento si los trabajos interactivos someten la petición de la transacción a un solo trabajo de proceso por lotes.

- Las colas de datos son el medio más rápido de comunicación asíncrona entre dos trabajos. La utilización de colas de datos para enviar y recibir datos requiere una menor actividad general que la utilización de archivos de base de datos, colas de mensajes o áreas de datos para enviar y recibir datos.

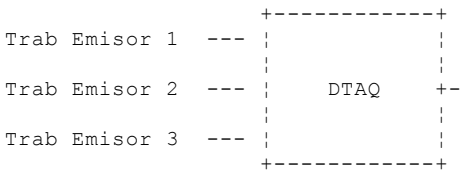
- Puede enviar, recibir y recuperar la descripción de una cola de datos en cualquier programa o procedimiento HLL llamando a los programas QSNDDTAQ, QRCVDTAQ, QMHRDQM, QCLRDTAQ y QMHQRDQD sin salir del programa o procedimiento HLL o llamando a un programa o procedimiento CL para enviar, recibir, borrar o recuperar la descripción.

- Al recibir datos de una cola de datos, puede establecer un tiempo de espera para que el trabajo espere hasta que llegue una entrada en la cola de datos. Esto es distinto de la utilización del parámetro EOFDLY en el mandato OVRDBF, que hace que se active un trabajo cuando se sobrepasa el tiempo de retardo.

- Más de un trabajo puede recibir datos de la misma cola de datos, lo que es una ventaja en determinadas aplicaciones en las que el número de entradas que se van a procesar es mayor que las que puede manejar un trabajo dentro de las limitaciones de rendimiento deseadas. Por ejemplo, si varias impresoras están disponibles para imprimir órdenes, varios trabajos interactivos podrían enviar peticiones a una sola cola de datos. Un trabajo distinto para cada impresora podría podría recibir datos de la cola de datos, ya sea en orden primero en entrar, primero en salir (FIFO), último en entrar, primero en salir (LIFO) o en orden de cola según la clave.

- Las colas de datos ofrecen la posibilidad de adjuntar un ID de emisor a cualquier mensaje que se encuentre en la cola. El ID de emisor, un atributo de la cola de datos que se establece al crear la propia cola, contiene el nombre de trabajo calificado, así como el perfil del usuario actual.

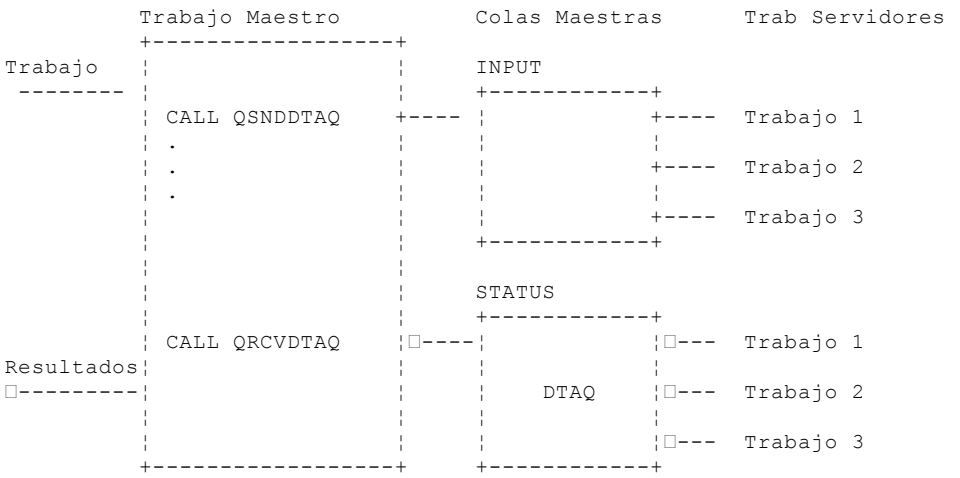
A continuación se facilita un ejemplo en el que se muestra cómo funcionan las colas de datos. Varios trabajos colocan entradas en una cola de datos. Las entradas las maneja un trabajo servidor. Esto puede utilizarse para que varios trabajos envíen órdenes procesadas a un sólo trabajo que realizaría la impresión. Puede enviarse un número ilimitado de trabajos a la misma cola.



A continuación figura otro ejemplo que utiliza colas de datos. Un trabajo principal obtiene las órdenes de trabajo y envía las entradas a una cola de datos (llamando al programa QSNDDTAQ). Los trabajos servidores reciben reciben las entradas de la cola de datos (llamando al programa QRCVDTAQ) y procesan los datos. Los trabajos servidores pueden devolver el informe de estado al programa principal utilizando otra cola de datos.

Las colas de datos permiten que el trabajo principal dirija el trabajo hacia los trabajos servidores. Esto evita que el trabajo principal tenga que recibir la siguiente petición de trabajo. Cualquier número de

trabajos servidores pueden recibir datos de la misma cola de datos.



Cuando no hay entradas en una cola de datos, los trabajos servidores tienen las siguientes opciones:

- Esperar hasta que haya una entrada situada en la cola
- Esperar durante un periodo de tiempo determinado; si, una vez transcurrido este tiempo, la entrada aún no ha llegado, el proceso continúa
- No esperar, volver inmediatamente.

Las colas de datos también se pueden utilizar cuando un programa necesita esperar entrada de los archivos de pantalla, archivos ICF y colas de datos al mismo tiempo. Cuando especifica el parámetro DTAQ para los siguientes mandatos:

- Crear Archivo de Pantalla (CRTDSPF)
- Cambiar Archivo de Pantalla (CHGDSPF)
- Alterar Temporalmente Archivo de Pantalla (OVRDSPF)
- Crear Archivo ICF (CRTICFF)
- Cambiar Archivo ICF (CHGICFF)
- Alterar Temporalmente Archivo ICF (OVRICFF)

puede indicar una cola de datos que contendrá entradas cuando suceda una de las siguientes acciones:

- Se pulsa una tecla de mandato habilitada o la tecla Intro desde un dispositivo de pantalla invitado

- Los datos se hacen disponibles desde una sesión ICF invitada

Las colas de datos también pueden asociarse con colas de salida. Consulte CL Reference para detalles sobre el mandato Crear Cola de Salida (CRTOUTQ).

Los trabajos que se ejecutan en el sistema también pueden colocar entradas en la misma cola de datos que la especificada en el parámetro DTAQ mediante el programa QSNDDTAQ.

Una aplicación llama al programa QRCVDTAQ para recibir cada entrada colocada en la cola de datos y después procesa la entrada basándose en si la colocó aquí un archivo de pantalla, un archivo ICF o el programa QSNDDTAQ. Para obtener más información, véanse los apartados "Ejemplo 2: Espera de una Entrada de un Archivo de Pantalla y de un Archivo ICF" en el tema 3.5.7 y "Ejemplo 3: Espera de una Entrada de un Archivo de Pantalla y de una Cola de Datos" en el tema 3.5.7.

Subtemas

3.5.1 Colas de Datos Remotas

Puede acceder a colas de datos remotas con los archivos de Gestión de Datos Distribuidos (DDM). Los archivos DDM lo hacen posible para un programa residente en un AS/400 para acceder a una cola de datos en un AS/400 remoto, para realizar cualquiera de las funciones siguientes:

- enviar datos a una cola de datos
- recibir datos de una cola de datos
- borrar datos de una cola de datos

Un programa de aplicación que utilice habitualmente una cola de datos estándar, también puede acceder a una cola de datos DDM remota, sin que sea necesario cambiar o compilar de nuevo la aplicación. Para asegurarse de que se accede a la cola de datos correcta, debe realizar una de las siguientes acciones:

- Eliminar la cola de datos estándar y crear una cola de datos DDM que tenga el mismo nombre que la cola de datos estándar original.
- Renombrar la cola de datos estándar.

Puede crear la cola de datos DDM con el siguiente mandato:

```
      CRTDTAQ DTAQ(LOCALLIB/DDMDTAQ) TYPE(*DDM)
         RMTDTAQ(REMOTELIB/REMOTEDTAQ) RMTLOCNAME(SISTEMAB)
         TEXT('cola de datos DDM para acceder a cola de datos en SISTEMAB')
```

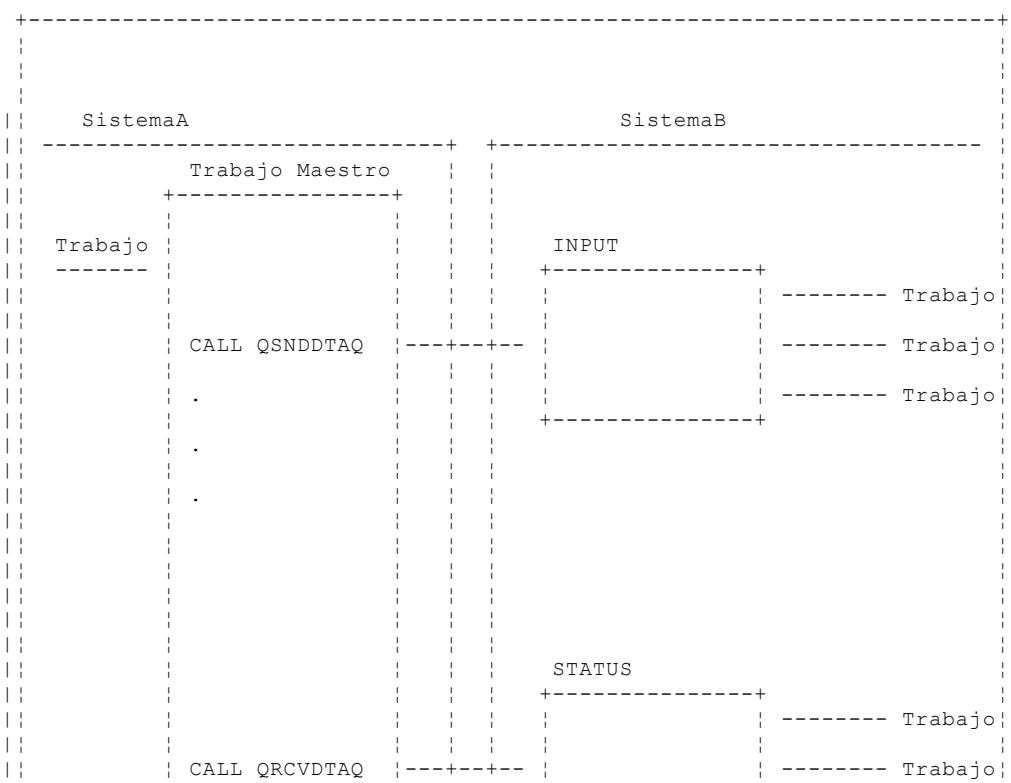
También puede utilizar una expansión del ejemplo anterior (trabajo Maestro/Servidor) para crear una cola de datos DDM, para utilizar con colas de datos remotas. El trabajo maestro reside en el SistemaA; la cola de datos y los trabajos servidores se mueven al SistemaB. Después de crear dos colas de datos DDM (INPUT y STATUS), el trabajo maestro continúa con la comunicación asíncronamente con los trabajos servidores que residen en el SistemaB. El siguiente ejemplo muestra la manera de crear una cola de datos DDM con colas de datos remotas:

```
      CRTDTAQ DTAQ(LOCALLIB/INPUT) TYPE(*DDM)
         RMTDTAQ(REMOTELIB/INPUT) RMTLOCNAME(SistemaB)
         TEXT('cola de datos DDM para acceder a INPUT en SISTEMAB')

      CRTDTAQ DTAQ(LOCALLIB/STATUS) TYPE(*DDM)
         RMTDTAQ(REMOTELIB/STATUS) RMTLOCNAME(SistemaB)
         TEXT('cola de datos DDM para acceder a STATUS en SISTEMAB')
```

El trabajo maestro llama a QSNDDTAQ, entonces pasa el nombre de la cola de datos de LOCALLIB/INPUT y envía los datos a una cola de datos remota (REMOTELIB/INPUT) en el SistemaB. Para recibir datos de la cola de datos remota, (REMOTELIB/STATUS), el trabajo maestro pasa el nombre de la cola de datos a LOCALLIB/STATUS para la llamada a QRCVDTAQ.

Consulte el manual CL Reference o la publicación *Gestión de datos* para obtener mas información sobre colas de datos DDM.



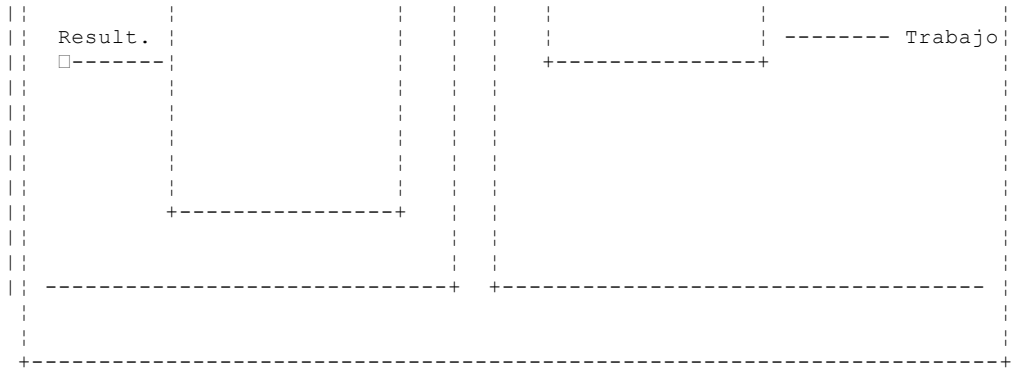


Figura 3-1. Ejemplo de Acceso a una Cola de Datos Remota

3.5.2 Comparaciones en la Utilización de Archivos de Base de Datos como Colas

A continuación se describen las diferencias que existen entre el uso de colas de datos y archivos de base de datos:

- Las colas de datos se han mejorado para la comunicación entre programas y procedimientos activos, no para almacenar grandes volúmenes de datos o un gran número de entradas. Para ello, utilice archivos de base de datos como colas.

- Las colas de datos no deben utilizarse para el almacenamiento a largo plazo de los datos. Para ello, deben utilizarse los archivos de base de datos.

- Al utilizar colas de datos, incluya rutinas de finalización anómala en sus programas para recuperar las entradas no procesadas por completo antes de que finalice el sistema.

- Es aconsejable borrar y volver a crear una cola de datos en un lugar seguro periódicamente (por ejemplo, una vez al día). La existencia de demasiadas entradas sin borrar puede afectar al rendimiento. El hecho de volver a crear la cola de datos de forma periódica hará que la cola de datos vuelva a tener el tamaño óptimo.

3.5.3 Similitudes con las Colas de Mensajes

Las colas de datos se parecen a las colas de mensajes en el hecho de que los procedimientos y programas pueden enviar datos a la cola, que recibe más tarde otro procedimiento o programa. Sin embargo, aunque puede haber varios programas que tengan una recepción pendiente en una cola de datos al mismo tiempo, sólo uno puede tener una recepción pendiente en una cola de mensajes en un momento determinado (sólo un programa recibe una entrada de una cola de datos, aunque haya más de un programa esperando). Las entradas de una cola de datos se manejan en orden primero en entrar primero en salir, último en entrar primero en salir o en orden de clave. Cuando se recibe una entrada, se suprime de la cola de datos.

3.5.4 Prerrequisitos para la Utilización de Colas de Datos

Antes de usar una cola de datos, primero debe crearla utilizando el mandato Crear Cola de Datos (CRTDTAQ). A continuación se facilita un ejemplo:

```
  CRTDTAQ DTAQ(MYLIB/INPUT)  MAXLEN(128)
          TEXT('Cola de datos de muestra')
```

El parámetro MAXLEN es obligatorio, y especifica la longitud máxima (1 a 64 512 caracteres) de las entradas que pueden enviarse a la cola de datos.

3.5.5 Gestión del Almacenamiento Utilizado por una Cola de Datos

Cuando se recibe una entrada de una cola de datos, la entrada se elimina de la cola de datos pero el almacenamiento auxiliar no se libera. El mismo almacenamiento auxiliar se utiliza de nuevo cuando se envía una nueva entrada a la cola de datos. La cola aumenta a medida que se envían las entradas a la cola y no se reciben. El rendimiento mejora si el tamaño de la cola se mantiene en menos de 100 entradas. Si una cola de datos aumenta demasiado, suprímala mediante el mandato Suprimir Cola de Datos (DLTDTAQ) y vuelva a crearla utilizando el mandato Crear Cola de Datos (CRTDTAQ).

3.5.6 Asignación de Colas de Datos

Si la aplicación requiere que no tenga acceso a una cola de datos más de un trabajo al mismo tiempo, debe codificarla de manera que incluya un mandato Asignar Objeto (ALCOBJ) antes de utilizar una cola de datos. La cola de datos debe desasignarse mediante el mandato Desasignar Objeto (DLCOBJ) cuando la aplicación haya terminado de utilizarla.

El mandato ALCOBJ *no* limita, por sí solo, el que otro trabajo envíe o reciba datos de una cola de datos o borre una cola de datos. Sin embargo, si todas las aplicaciones están codificadas para incluir el mandato ALCOBJ antes de utilizar una cola de datos, la asignación de una cola de datos ya asignada a otro trabajo dará error, evitando la utilización de la cola de datos por más de un trabajo al mismo tiempo.

Cuando una asignación falla porque la cola de datos ya está asignada a otro trabajo, el sistema emite un mensaje de error, CPF1002. Puede utilizarse el mandato Supervisar Mensaje (MONMSG) en el procedimiento de la aplicación para supervisar este mensaje y responder a un mensaje de error. Las posibles respuestas incluyen el envío de un mensaje al usuario y el intento de asignar otra vez la cola de datos. Véase el apartado " Supervisión de Mensajes en un Programa o Procedimiento CL" en el tema 8.3 para obtener más información.

3.5.7 Ejemplos de Utilización de Colas de Datos

En los siguientes ejemplos se explican tres métodos distintos para procesar los archivos de cola de datos.

Ejemplo 1: Esperar dos horas como máximo para recibir datos de la cola de datos: En el siguiente ejemplo, en el programa B se especifica una espera de dos horas como máximo (7200 segundos) para recibir una entrada de la cola de datos. El programa A envía una entrada a la cola de datos DTAQ1 situada en la biblioteca QGPL. Si el programa A envía una entrada en un periodo de dos horas, el programa B recibe las entradas de esta cola de datos. El proceso empieza inmediatamente. Si transcurren dos horas sin que el programa A envíe una entrada, el programa B procesa la condición de tiempo excedido porque la longitud de campo devuelta es 0. El programa B continúa recibiendo entradas hasta que se da esta condición de tiempo excedido. Los programas están escritos en CL; sin embargo, todos los programas se pueden escribir en cualquier lenguaje de alto nivel.

La cola de datos se crea con el siguiente mandato:

```
CRTDTAQ DTAQ(QGPL/DTAQ1) MAXLEN(80)
```

En este ejemplo, todas las entradas de la cola de datos tienen una longitud de 80 bytes.

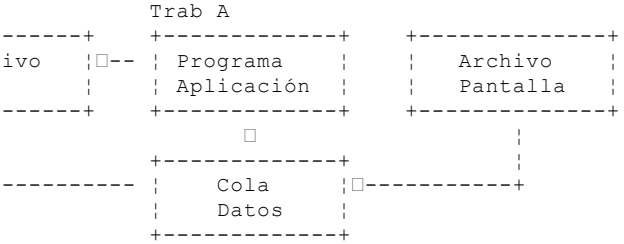
En el programa A, las sentencias siguientes se refieren a la cola de datos:

```
PGM
DCL  &FLDLEN  *DEC  LEN(5 0)  VALUE(80)
DCL  &FIELD  *CHAR  LEN(80)
.
.(determinar datos a enviar a la cola)
.
CALL  QSNDDTAQ  PARM(DTAQ1 QGPL &FLDLEN &FIELD)
.
.
.
```

En el programa B, las sentencias siguientes se refieren a la cola de datos:

```
         PGM
         DCL  &FLDLEN  *DEC  LEN(5 0)  VALUE(80)
         DCL  &FIELD  *CHAR  LEN(80)
         DCL  &WAIT  *DEC  LEN(5 0)  VALUE(7200)   /* 2 horas */
         .
         .
         .
LOOP:    CALL QRCVDTAQ  PARM(DTAQ1 QGPL &FLDLEN &FIELD &WAIT)
         IF    (&FLDLEN *NE 0) DO   /* Entrada recibida */
                .
                . (procesar datos de cola de datos)
                .
                GOTO LOOP   /* Obtener siguiente entrada de cola datos */
         ENDDO
         .
         . (no se han recibido entradas en 2 horas; procesar condición
         .  tiempo espera excedido)
         .
```

Ejemplo 2: Espera de una Entrada de un Archivo de Pantalla y de un Archivo ICF: Este ejemplo es diferente de la utilización normal de las colas de datos, ya que sólo hay un trabajo. La cola de datos actúa como un objeto de comunicación dentro del trabajo en lugar de hacerlo entre dos trabajos.



En este ejemplo, un programa espera un entrada de un archivo de pantalla y

de un archivo ICF. En lugar de esperar alternativamente una entrada y después la otra, se utiliza una cola de datos para permitir que el programa espere en un objeto (la cola de datos). El programa llama a QRCVDTAQ y espera que una entrada se sitúe en la cola de datos que se especificó en el archivo de pantalla y en el archivo ICF. Ambos archivos especifican la misma cola de datos. El soporte de gestión de datos de pantalla y de datos ICF colocan dos tipos de entrada en la cola cuando los datos están disponibles en cualquiera de los dos archivos. Las entradas de archivo ICF comienzan con ***ICFF** y las entradas de archivo de pantalla comienzan con ***DSPF**.

La entrada de archivo de pantalla o de archivo ICF que se sitúa en la cola de datos tiene una longitud de 80 caracteres y contiene los atributos de campo descritos en la lista siguiente. Por lo tanto, la cola de datos que se especifica mediante los mandatos CRTDSPF, CHGDSPF, OVRDSPF, CRTICFF, CHGICFF y OVRICFF debe tener una longitud mínima de 80 caracteres.

Posición (y tipo de datos)	**Descripción**
1 a 10 (carácter)	Tipo de archivo que situó la entrada en la cola de datos. Este campo tendrá uno de estos dos valores: ***ICFF** para el archivo ICF ***DSPF** para el archivo de pantalla Si el trabajo que recibe los datos de la cola de datos tiene solamente un archivo de pantalla o un archivo ICF abierto, entonces éste es el único archivo necesario para determinar qué tipo de entrada se ha recibido de la cola de datos.
11 a 12 (binario)	Identificador exclusivo para el archivo. El valor del identificador es el mismo que el valor del área de realimentación de apertura del archivo. El programa que recibe la entrada de la cola de datos debe utilizar este campo sólo si hay más de un archivo con el mismo nombre que sitúe entradas en la cola de datos.
13 a 22 (carácter)	Nombre del archivo de pantalla o archivo ICF. Es el nombre del archivo abierto en ese momento, después de que se hayan procesado todas las alteraciones temporales; es el mismo que el nombre del archivo del área de realimentación de apertura del archivo. El programa que recibe la entrada de la cola de datos debe utilizar este campo sólo si hay más de un archivo de pantalla o ICF que coloca entradas en la cola de datos.
23 a 32 (carácter)	Biblioteca en la que se encuentra el archivo. Es el nombre de la biblioteca después de que se hayan procesado todas las alteraciones temporales; es el mismo que el nombre de biblioteca que se ha encontrado en el área de realimentación de apertura del archivo. El programa que recibe la entrada de la cola de datos debe utilizar este campo sólo si hay más de un archivo de pantalla o ICF que coloca entradas en la cola de datos.
33 a 42 (carácter)	Nombre del dispositivo de programa, después de que se hayan procesado todas las alteraciones temporales. Es el mismo que el nombre que se ha encontrado en la lista de definiciones de dispositivos de programa del área de realimentación de apertura. Para el tipo de archivo *DSPF, éste es el nombre del dispositivo de pantalla en el que se utilizó el mandato o la tecla Intro. Para el tipo de archivo *ICFF, es el nombre del dispositivo de programa en que hay datos disponibles. El programa que recibe la entrada de la cola de datos debe utilizar este campo sólo si el archivo que situó la entrada en la cola de datos tiene más de un dispositivo o sesión invitados antes de la recepción de la entrada de cola de datos.
43 a 80 (carácter)	Reservado.

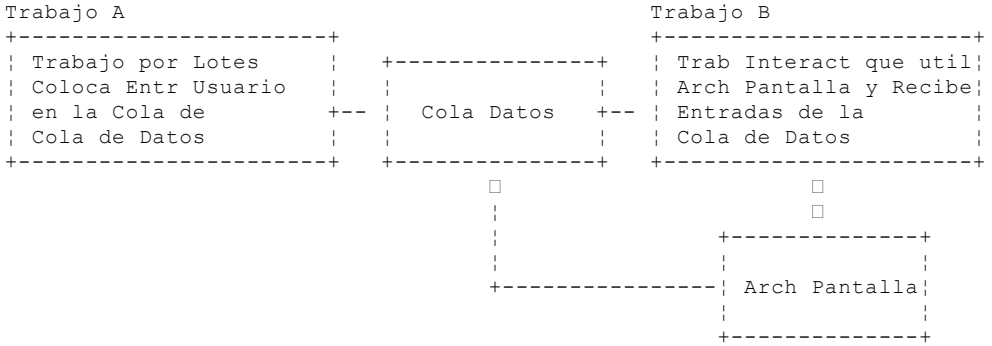
El siguiente ejemplo muestra la lógica de decodificación que utilizaría el programa descrito anteriormente:

```
    .
    .
    OPEN DSPFILE ... /* Abrir archivo pantalla. Parám DTAQ especificado en */
                     /* CRTDSPF, CHGDSPF o OVRDSPF para el archivo.        */

    OPEN ICFFILE ... /* Abrir archivo ICF. Parámetro DTAQ especificado en  */
                     /* CRTICFF, CHGICFF o OVRICFF para el archivo.        */
   .
   .
   DO
     WRITE DSPFILE   /* Escribir con Invitación para archivo de pantalla   */
     WRITE ICFFILE   /* Escribir con Invitación para el archivo ICF        */

     CALL QRCVDTAQ   /* Recibir entrada de cola de datos especificada en  */
                     /* parámetros DTAQ para los archivos. Las entradas   */
                     /* se colocan en la cola de datos cuando los datos   */
                     /* están disponibles desde cualquier disp o sesión   */
                     /* invitados en cualquier archivo.                   */
                     /* Tras recibir la entrada, determine qué archivo    */
                     /* tiene datos disponibles, léalos, procéselos,      */
                     /* invite de nuevo el archivo y vuelva a procesar    */
                     /* la siguiente entrada de la cola de datos.         */
     IF 'ENTRY TYPE' FIELD = '*DSPF     ' THEN   /* Entrada de archivo de */
       DO                                       /*  pantalla. Esta entrada*/
                                                /* no contiene los datos  */
                                                /* recibidos, por tanto   */
                                                /* los datos debe leerse  */
                                                /* del archivo antes de   */
         READ DATA FROM DISPLAY FILE            /* procesarlos.           */
         PROCESS INPUT DATA FROM DISPLAY FILE
         WRITE TO DISPLAY FILE                   /* Escribir con Invitac. */
       END
     ELSE                                       /* La entrada es del arch */
                                                /* ICF. Esta no contiene  */
                                                /* los datos recibidos,   */
                                                /* por lo tanto los datos */
                                                /* deben leerse del       */
                                                /* archivo antes de poder */
                                                /* procesarlos.           */
         READ DATA FROM ICF  FILE
         PROCESS INPUT DATA FROM ICF FILE
         WRITE TO ICF FILE                      /* Escribir con Invitac. */
     LOOP BACK TO RECEIVE ENTRY FROM DATA QUEUE
   .
   .
   END
```

Ejemplo 3: Espera de una Entrada de un Archivo de Pantalla y de una Cola de Datos: En el ejemplo siguiente, el programa en el Trabajo B espera la entrada de un archivo de pantalla que está utilizando y la llegada de entrada a la cola de datos del Trabajo A. En lugar de esperar alternativamente el archivo de pantalla y luego la cola de datos, el programa espera un objeto, la cola de datos.



El programa llama a QRCVDTAQ y espera que una entrada se sitúe en la cola de datos que se especificó en el archivo de pantalla. El trabajo A también sitúa entradas en la misma cola de datos. Existen dos tipos de entradas situadas en esta cola, la entrada del archivo de pantalla y la entrada definida por el usuario. La gestión de datos de pantalla coloca la entrada del archivo de pantalla en la cola de datos cuando hay datos disponibles

de dicho archivo. El trabajo A sitúa la entrada definida por el usuario en la cola de datos.

La estructura de la entrada del archivo de pantalla se describe en el ejemplo anterior.

La estructura de la entrada que el Trabajo A coloca en la cola la define el programador de la aplicación.

El siguiente ejemplo muestra la lógica de codificación que el programa de aplicación en el Trabajo B utilizaría:

```
    .
    .
    .
    .
    OPEN DSPFILE ... /* Abrir archivo pantalla. Parám DTAQ especificado en */
                     /* CRTDSPF, CHGDSPF o OVRDSPF para el archivo.        */


   .
   .
   DO
     WRITE DSPFILE   /* Escribir con Invitación para archivo de pantalla   */

     CALL QRCVDTAQ   /* Recibir entrada de cola de datos especificada en  */
                     /* parámetros DTAQ para el archivo. Las entradas     */
                     /* se colocan en la cola de datos por Trab A o por   */
                     /* gestión datos pantalla cuando los datos están     */
                     /* disponibles desde cualquier dispositivo invitado  */
                     /* en el archivo de pantalla.                        */
                     /* Tras recibir la entrada, determine qué tipo       */
                     /* de entrada es, procésela y vuelva a recibir la    */
                     /* siguiente entrada de la cola de datos.            */
     IF 'ENTRY TYPE' FIELD = '*DSPF     ' THEN   /* Entrada de archivo de */
       DO                                       /*  pantalla. Esta entrada*/
                                                /* no contiene los datos  */
                                                /* recibidos, por tanto   */
                                                /* deben leerse del       */
                                                /* archivo antes de poder */
         READ DATA FROM DISPLAY FILE            /* procesarlos.           */
         PROCESS INPUT DATA FROM DISPLAY FILE
         WRITE TO DISPLAY FILE                   /* Escribir con Invitac. */
       END
     ELSE                                       /* Entrada de Trabajo A. */
                                                /* Esta entrada contiene */
                                                /* los datos del Trab A, */
                                                /* por lo que no necesita*/
                                                /* leerlos antes de      */
                                                /* procesar los datos.   */
         PROCESS DATA QUEUE ENTRY FROM JOB A
     LOOP BACK TO RECEIVE ENTRY FROM DATA QUEUE
   .
   .
   .
   END
```

3.6 Utilización de Áreas de Datos para la Comunicación entre Procedimientos y Programas

Un área de datos es un objeto que contiene datos a los que puede acceder cualquier trabajo que se procese en el sistema. Un área de datos puede utilizarse siempre que necesite almacenar un volumen limitado de información, independientemente de la existencia de procedimientos o archivos. Las áreas de datos se utiliza normalmente para:

- Proporcionar un área (quizás dentro de cada biblioteca QTEMP de trabajo) para pasar información dentro de un trabajo.

- Proporcionar un campo que se cambie fácil y frecuentemente para controlar referencias dentro de un trabajo, tales como:

  - Suministro del siguiente número de orden que deba asignarse
  - Suministro del siguiente número de comprobación
  - Suministro del siguiente volumen de soporte salvar/restaurar a utilizar

- Proporcionar un campo constante para utilizar en varios trabajos, tal como una cuota o lista de distribución.

- Proporcionar acceso limitado a un proceso más largo que requiere el área de datos. Un área de datos puede bloquearse para un único usuario, evitando así que otros usuarios procesen al mismo tiempo.

Para crear un área de datos diferente del área de datos local o de grupo, utilice el mandato Crear Área de Datos (CRTDTAARA). De este modo, creará un objeto separado en una biblioteca específica, y podrá inicializarlo con un valor. Si desea utilizar el valor en un programa o procedimiento CL, utilice el mandato Recuperar Área de Datos (RTVDTAARA) para transferir el valor actual a una variable de su procedimiento o programa. Si cambia este valor en su programa o procedimiento CL y desea devolver el valor nuevo al área de datos, utilice el mandato Cambiar Área de Datos (CHGDTAARA).

Para visualizar el valor actual, utilice el mandato Visualizar Área de Datos (DSPDTAARA). Puede suprimir un área de datos utilizando el mandato Suprimir Área de Datos (DLTDTAARA).

Subtemas

3.6.1 Área de Datos Local

Se crea un área de datos local para cada trabajo del sistema, incluyendo los trabajos de arranque automático, los trabajos arrancados en el sistema por un lector y los trabajos de supervisión del subsistema.

El sistema crea un área de datos local, que inicialmente se rellena con blancos, con una longitud de 1024 y de tipo *CHAR. Al someter un trabajo utilizando el mandato SBMJOB, el valor del área de datos local del trabajo que somete se copia en el área de datos local del trabajo sometido. Puede hacer referencia al área de datos local de su trabajo especificando *LDA en la palabra clave DTAARA de los mandatos CHGDTAARA, RTVDTAARA y DSPDTAARA o *LDA en la función incorporada de subserie (%SST).

En cuanto al área de datos local son ciertas las siguientes afirmaciones:

- No puede hacerse referencia al área de datos local desde otro trabajo.
- No puede crear, suprimir ni asignar un área de datos local.
- No hay ninguna biblioteca asociada con el área de datos local.

El contenido del área de datos local existe a través de los límites de los pasos de direccionamiento. Por lo tanto, la utilización de los mandatos Transferir Trabajo (TFRJOB), Transferir Trabajo por Lotes (TFRBCHJOB), Redireccionar Trabajo (RRTJOB) o Regresar (RETURN) no afecta al contenido del área de datos local.

El área de datos local puede utilizarse para:

- Pasar información a un procedimiento o programa sin utilizar la lista de parámetros.
- Pasar información a un trabajo sometido cargando la información en el área de datos local y sometiendo el trabajo. A continuación, se puede acceder a los datos desde el trabajo sometido.
- Mejorar el rendimiento sobre otros tipos de acceso al área de datos desde un programa o procedimiento CL.
- Almacenar información sin la actividad adicional de crear y suprimir uno mismo un área de datos.

La mayoría de lenguajes de alto nivel pueden utilizar también el área de datos local. Los mandatos SBMxxxJOB y STRxxxRDR hacen que los trabajos se arranquen con un área de datos local inicializada con espacios en blanco. Sólo el mandato SBMJOB permite que el contenido del área de datos local del trabajo que somete se pase al nuevo trabajo.

3.6.2 Área de Datos de Grupo

El sistema crea un área de datos de grupo cuando un trabajo interactivo se convierte en un trabajo de grupo (mediante el mandato Cambiar Atributos de Grupo ([CHGGRPA]). Sólo puede existir un área de datos de grupo por cada grupo. El área de datos de grupo se borra al finalizar el último trabajo en el grupo (mediante los mandatos ENDJOB, SIGNOFF o ENDGRPJOB o con un final anormal), o bien cuando el trabajo ya no forma parte del trabajo de grupo (mediante el mandato CHGGRPA con la especificación de GRPJOB(*NONE)).

Un área de datos de grupo, que se rellena inicialmente con espacios en blanco, tiene una longitud de 512 y es de tipo *CHAR. Puede utilizar un área de datos de grupo desde un trabajo de grupo especificando *GDA para el parámetro DTAARA en los mandatos CHGDTAARA, RTVDTAARA y DSPDTAARA. Un área de datos de grupo es accesible para todos los trabajos del grupo.

Algunas de las características del área de datos de grupo son:

- No puede utilizar el área de datos de grupo como un sustituto para una variable de tipo carácter en la función incorporada de subserie (%SUBSTRING o %SST). Sin embargo, puede entrar o extraer una variable de tipo carácter de 512 bytes utilizada por la función de subserie del área de datos de grupo.

- Los trabajos que están fuera del grupo no pueden hacer referencia a un área de datos de grupo.

- No puede crear, suprimir ni asignar un área de datos de grupo

- No hay ninguna biblioteca asociada con el área de datos de grupo.

El mandato Transferir a Trabajo de Grupo (TFRGRPJOB) no cambia el contenido de un área de datos de grupo.

Además de utilizar el área de datos de grupo igual que se utilizan otras áreas de datos, puede emplearla para comunicar información entre trabajos del mismo grupo. Por ejemplo después de emitir el mandato Cambiar Atributos de Trabajo de Grupo (CHGGRPA), puede utilizarse el mandato siguiente para establecer el valor del área de datos de grupo:

```
  CHGDTAARA  DTAARA(*GDA)  VALUE('Enero1988')
```

este mandato se puede ejecutar desde un programa o puede emitirlo el usuario de la estación de trabajo.

Cualquier otro programa o procedimiento CL del grupo puede recuperar el valor del área de datos de grupo con el siguiente mandato CL:

```
  RTVDTAARA  DTAARA(*GDA)  RTNVAR(&GRPARA)
```

Este mandato coloca el valor del área de datos de grupo (**Enero1988**) en la variable CL &GRPARA.

3.6.3 Área de datos del Parámetro de Inicialización del Programa (PIP)

Cuando se inicia el trabajo, se crea un área de datos PIP (PDA) por cada trabajo de pre-arranque. El sub-tipo de objeto del PDA es diferente a un área de datos regulares. Al PDA sólo se le puede asignar el nombre de valor especial *PDA. El tamaño del PDA es de 2000 bytes, pero puede contener un número indeterminado de parámetros.

Los mandatos CL RTVDTAARA, CHGDTAARA, y DSPDTAARA y las macroinstrucciones RTVDTAARA y CHGDTAARA dan soporte al valor especial *PDA para el parámetro de nombre de área de datos.

3.6.4 Áreas de Datos Remotas

Puede acceder a áreas de datos remotas utilizando Gestión de Datos Distribuidos (DDM). Si un programa de aplicación que reside en un AS/400 modifica o recupera un área de datos que reside en un AS/400 remoto, no es necesario que este programa se modifique o recompile de nuevo. Para asegurarse de que se accede a la cola de datos correcta, debe realizar una de las siguientes acciones:

- Eliminar el área de datos estándar y crear un área de datos DDM que tenga el mismo nombre que el área de datos original.
- Redenominar el área de datos estándar.

Puede crear un área de datos DDM haciendo lo siguiente:

```
      CRTDTAARA DTAARA(LOCALLIB/DDMDTAARA) TYPE(*DDM)
         RMTDTAARA(REMOTELIB/RMTDTAARA) RMTLOCNAME(SISTEMAB)
         TEXT('Área de datos DDM para acceder a área de datos en SISTEMAB')
```

Para utilizar en un programa CL un valor de un área de datos en un AS/400 remoto, utilice el mandato Recuperar Área de Datos (RTVDTAARA). Especifique el nombre de un área de datos DDM para traer el valor actual a una variable en su programa. Si cambia este valor en su programa CL y desea devolver este nuevo valor al área de datos remota, utilice el mandato Cambiar Área de Datos (CHGDTAARA) y especifique la misma área de datos DDM.

Si especifica el nombre de un área de datos DDM al utilizar el mandato Visualizar Área de Datos (DSPDTAARA), se visualiza el valor del área de datos DDM, en lugar del valor del área de datos remota. Puede suprimir un área de datos DDM utilizando el mandato Suprimir Área de Datos (DLTDTAARA).

Consulte el manual CL Reference y la publicación *Gestión de datos* para obtener mas información acerca de áreas de datos DDM.

3.6.5 Creación de un Área de Datos

A diferencia de las variables, las áreas de datos son objetos y deben crearse antes de poder utilizarlos. Un área de datos se puede crear como:

- Una serie de caracteres que puede tener una longitud de 2000 caracteres.

- Un valor decimal con atributos diferentes, dependiendo de si va a utilizarse solamente en un programa o procedimiento CL o también con otros programas o procedimientos de lenguaje de alto nivel. Para programas y procedimientos CL, el área de datos puede tener como máximo 15 dígitos a la izquierda de la coma decimal y como máximo 9 dígitos a la derecha, pero sólo 15 en total. Para otros lenguajes, el área de datos puede tener hasta 15 dígitos a la izquierda de la coma decimal y hasta 9 a la derecha, para un total de hasta 24 dígitos.

- Un valor lógico '0' ó '1', donde '0' puede significar desactivado, falso o no, y '1' puede significar activado, verdadero o sí.

Al crear un área de datos, también puede especificar un valor inicial para la misma. Si no especifica ninguno, se supone lo siguiente:

- 0 para decimal.
- Espacios en blanco para caracteres.
- '0' para lógico.

Para crear un área de datos, utilice el mandato Crear Área de Datos (CRTDTAARA). En el ejemplo siguiente se crea un área de datos para pasar el número de un cliente de un programa a otro:

```
  CRTDTAARA  DTAARA(CUST) TYPE(*DEC) +
             LEN(5 0) TEXT('Siguiente número de cliente')
```

3.6.6 Bloqueo y Asignación del Área de Datos

El mandato CHGDTAARA utiliza un bloqueo *SHRUPD (compartido para actualización) en el área de datos durante el proceso del mandato. Los mandatos RTVDTAARA y DSPDTAARA utilizan un bloqueo *SHRRD (compartido para lectura) en el área de datos durante el proceso del mandato. Si se realiza más de una operación en un área de datos, puede desear utilizar el mandato Asignar Objeto (ALCOBJ) para impedir que otros usuarios accedan al área de datos hasta que no hayan finalizado las operaciones. Por ejemplo, si el área de datos contiene un valor que los trabajos que se ejecutan al mismo tiempo leen y aumentan, puede utilizarse el mandato ALCOBJ para proteger el valor en las operaciones de lectura y de actualización. Véase el Capítulo 4 para obtener información sobre cómo asignar objetos.

Para obtener información sobre el manejo de áreas de datos en otros lenguajes (no en CL), remítase al manual de consulta de HLL adecuado.

3.6.7 Visualización de un Área de Datos

Puede visualizar los atributos (nombre, biblioteca, tipo, longitud y descripción del texto del área de datos) y el valor de un área de datos. Consulte el manual CL Reference para una detallada descripción del mandato Visualizar Área de Datos (DSPDTAARA).

La pantalla utiliza el formato de 24 dígitos con ceros iniciales suprimidos.

3.6.8 Cambio de un Área de Datos

El mandato Cambiar Área de Datos (CHGDTAARA) cambia la totalidad o parte del valor de un área de datos determinada. No cambia ningún otro atributo del área de datos. El valor nuevo puede ser una constante o una variable CL. Si el mandato está en un procedimiento CL, el área de datos no es necesario que exista cuando se crea el programa.

3.6.9 Recuperación de un Área de Datos

El mandato Recuperar Área de Datos (RTVDTAARA) recupera la totalidad o parte de un área de datos especificada y la copia en una variable CL. No es necesario que el área de datos exista en el momento de compilar, y no es necesario que la variable CL tenga el mismo nombre que el área de datos. Observe que este mandato recupera el contenido del área de datos especificada sin alterarlo.

3.6.10 Ejemplos de Recuperación de un Área de Datos

Ejemplo 1: Suponga que utiliza un área de datos denominado ORDINFO hacer un seguimiento del estado de un archivo de pedidos. Este área de datos está diseñada para que:

- La posición 1 contenga una O (abrir), una P (procesar) o una C (completar).
- La posición 2 contenga una I (en stock) o una O (fuera de stock).
- Las posiciones 3 a 5 contengan las iniciales del empleado que toma los pedidos.

Estos campos se declararían en el procedimiento tal como se muestra a continuación:

```
  DCL VAR(&ORDSTAT) TYPE(*CHAR) LEN(1)
  DCL VAR(&STOCKC) TYPE(*CHAR) LEN(1)
  DCL VAR(&CLERK) TYPE(*CHAR) LEN(3)
```

Para recuperar el estado de los pedidos en &ORDSTAT, entraría lo siguiente:

```
  RTVDTAARA DTAARA(ORDINFO (1  1))  RTNVAR(&ORDSTAT)
```

Para recuperar la condición de stock en &STOCK, introduciría lo siguiente:

```
  RTVDTAARA DTAARA(ORDINFO (2  1)) RTNVAR(&STOCKC)
```

Para recuperar las iniciales del empleado en &CLERK, introduciría lo siguiente:

```
  RTVDTAARA DTAARA(ORDINFO (3  3)) RTNVAR(&CLERK)
```

Cada utilización del mandato RTVDTAARA requiere acceder al área de datos. Si se están recuperando muchos subcampos, es más eficaz recuperar el área de datos completa en una variable y luego utilizar la función incorporada de subserie para extraer los subcampos.

Ejemplo 2: El ejemplo siguiente del mandato RTVDTAARA coloca el contenido especificado del área de datos de 5 caracteres en una variable de 3 caracteres. En este ejemplo:

- Se crea un área de datos de 5 caracteres denominada DA1 (en la biblioteca MYLIB) con el valor inicial de **'ABCDE'**
- Se declara una variable de 3 caracteres denominada &CLVAR1
- Se copia el contenido de las últimas tres posiciones de DA1 en &CLVAR1

Para ello, se entran los mandatos siguientes:

```
  CRTDTAARA DTAARA(MYLIB/DA1) TYPE(*CHAR) LEN(5) VALUE(ABCDE)
  .
  .
  .
  DCL VAR(&CLVAR1) TYPE(*CHAR) LEN(3)
  RTVDTAARA DTAARA(MYLIB/DA1 (3 3)) RTNVAR(&CLVAR1)
```

&CLVAR1 ahora contiene **'CDE'**.

Ejemplo 3: El ejemplo siguiente del mandato RTVDTAARA coloca el contenido de un área decimal de 5 dígitos en una variable de 5 dígitos decimales. En este ejemplo:

- Se crea un área de datos de 5 dígitos denominada DA2 (en la biblioteca MYLIB) con dos posiciones decimales y el valor inicial de **12,39**
- Se declara una variable de 5 dígitos nombrada &CLVAR2, con una posición decimal.
- Se copia el contenido de DA2 en &CLVAR2

Para ello, se entran los mandatos siguientes:

```
  CRTDTAARA DTAARA(MYLIB/DA2) TYPE(*DEC) LEN(5 2) VALUE(12.39)
  .
  .
  .
  DCL  VAR(&CLVAR2) TYPE(*DEC) LEN(5 1)
  RTVDTAARA DTAARA(MYLIB/DA2) RTNVAR(&CLVAR2)
```

&CLVAR2 ahora contiene **0012.3** (se ha producido un truncamiento fraccional).

3.6.11 Ejemplo de Cambio y Recuperación de un Área de Datos

A continuación se facilita un ejemplo de utilización de los mandatos CHGDTAARA y RTVDTAARA para las operaciones de subserie de caracteres.

En este ejemplo:

- Se crea un área de datos de 10 caracteres denominada DA1 (en la biblioteca MYLIB) con el valor inicial de **ABCD5678IJ**
- Se declara una variable de 5 caracteres denominada &CLVAR1
- Se cambia el contenido del área de datos DA1 (desde la posición inicial 5 y por una longitud de 4) al valor **EFG** rellenando después de la G con un espacio en blanco)
- Se recupera el contenido del área de datos DA1 (desde la posición inicial de 5 por una longitud 5) en la variable CL &CLVAR1

Para ello, se entran los mandatos siguientes:

```
  DCL VAR(&CLVAR1) TYPE(*CHAR) LEN(5)
  .
  CRTDTAARA DTAARA(MYLIB/DA1) TYPE(*CHAR) LEN(10) +
       VALUE('ABCD5678IJ')
  .
  .
  .
  CHGDTAARA DTAARA((MYLIB/DA1) (5 4)) VALUE('EFG')
  RTVDTAARA DTAARA((MYLIB/DA1) (5 5)) RTNVAR(&CLVAR1)
```

La variable &CLVAR1 ahora contiene **'EFG I'**.

4.0 Capítulo 4. Objetos y Bibliotecas

Los objetos son las unidades básicas en las que los mandatos realizan las operaciones. Por ejemplo, tanto los programas como los archivos son objetos. Mediante los objetos podrá buscar, mantener y procesar los datos en el sistema AS/400. Mediante los objetos podrá encontrar, mantener y procesar sus datos en el sistema AS/400. Tan sólo necesita saber qué objeto y qué función (mandato) desea utilizar; no necesita saber la dirección del almacenamiento de sus datos si es que desea utilizar dicho objeto.

|**Nota:** Los objetos pueden residir en bibliotecas y directorios.
| (Anteriormente, un objeto solo podía residir en una biblioteca).
| Este capítulo contiene solo la información acerca de los objetos
| residentes en bibliotecas. Consulte el manual *Introducción al
| Sistema de Archivos Integrado*, SC10-9636 (SC41-4711), para obtener
| información específica acerca de directorios.

En este capítulo se incluye la Interfaz de Programación de Uso General y la Información de Ayuda Asociada.

Subtemas

4.1 Tipos de Objetos y Atributos Usuales

Cada tipo de objeto del sistema AS/400 tiene una única finalidad en el sistema y dispone de un conjunto de mandatos asociados que procesan este tipo de objeto. El manual *Programación - Resumen de consulta* contiene una lista completa de los tipos de objeto, las abreviaturas utilizadas como valores de los parámetros de tipo de objeto y la definición de objeto que pertenece a dicho tipo.

Cada tipo de objeto tiene una serie de atributos comunes que describen el objeto. Dichos atributos figuran en la Tabla 4-2 en el tema 4.7. La información de ayuda en línea correspondiente a la pantalla Visualizar Descripción de Objeto (DSPOBJD) describe tales atributos.

4.2 Funciones Realizadas sobre Objetos

Sobre los objetos se pueden realizar gran número de funciones. Algunas de ellas las realiza el propio sistema de manera automática, mientras que otras precisan una petición expresa del usuario mediante mandatos.

Subtemas
4.2.1 Funciones que Ejecuta el Sistema Automáticamente
4.2.2 Funciones que puede Realizar Utilizando Mandatos

4.2.1 Funciones que Ejecuta el Sistema Automáticamente

Las funciones realizadas automáticamente garantizan que los objetos se procesan de una forma coherente, segura y correcta. Estas funciones son:

- Verificación del tipo de objeto. El sistema comprueba el el tipo de objeto y el tipo de función que se realiza sobre el objeto para verificar que esa función pueda realizarse en ese tipo de objeto. Por ejemplo, si el objeto especificado en un mandato CALL no es un programa, la función de llamada no puede realizarse.

- Verificación de la autorización sobre el objeto. El sistema comprueba el objeto, la función y el usuario para verificar que éste último puede realizar esa función sobre ese objeto. Por ejemplo, si si USERA no está autorizado de ningún modo a utilizar OBJB, no puede solicitar la realización de ninguna de las funciones sobre el mismo.

- Refuerzo del bloqueo del objeto. El sistema se encarga de garantizar la integridad de los objetos cuando dos o más usuarios tratan de utilizar un objeto al mismo tiempo. Los cambios que se realizan simultáneamente en un objeto quedan bloqueados; los usuarios no pueden utilizar un objeto si éste se está modificando.

- Detección y notificación de un daño en el objeto. El sistema supervisa los errores durante el proceso de objetos y le comunica las anomalías que producen los contenidos irreconocibles de los objetos. Dichas anomalías le son comunicadas a través de mensajes estándares que manifiestan el daño en el objeto. El sistema está diseñado de tal manera que estas anomalías se producirán muy raramente, si bien su supervisión y comunicación proporciona seguridad.

4.2.2 Funciones que puede Realizar Utilizando Mandatos

Las funciones que puede solicitar a través de los mandatos son de dos tipos:

- Funciones específicas para cada tipo de objeto. Por ejemplo, crear, cambiar y visualizar. Las funciones específicas se describen en otros apartados de este manual que tratan de los tipos de objetos.

- Algunas funciones de uso común que se aplican a los objetos en general se tratan en esta guía:

Tabla 4-1. Funciones usuales para objetos

Función	**Página**
Búsqueda de varios o de un único objeto en una biblioteca	4.3.4
Especificación de autorización para objetos en una biblioteca	4.4.2
Colocación de objetos en bibliotecas	4.4.6
Descripción de objetos	4.6
Visualización de descripciones de objetos	4.7
Recuperación de descripciones de objetos	4.8
Detección de objetos no utilizados en el sistema	4.10
Traslado de objetos entre bibliotecas	4.11
Creación de objetos duplicados	4.12
Cambio de nombre de objetos	4.13
Supresión de objetos	4.15
Asignación y desasignación de objetos	4.16
Visualización de estados de bloqueo en objetos	4.16.1
Comprobación de la existencia de objetos	5.1.2

4.3 Bibliotecas

En el sistema AS/400, los objetos se agrupan en objetos especiales denominados *bibliotecas*. Los objetos se encuentran utilizando bibliotecas. Para acceder a un objeto en una biblioteca, debe tener autorización sobre ambos. Consulte la sección "Consideraciones de Seguridad para Objetos" en el tema 4.4.3 y la sección "Especificación de Autorización para Bibliotecas" en el tema 4.4.2 para obtener más información.

Si especifica un nombre de biblioteca en el mismo parámetro que el nombre de objeto, éste último se denomina *nombre calificado*. Por ejemplo, si entra un mandato en el que debe especificar un nombre calificado el nombre del objeto podría ser:

```
  DISTLIB/ORD040C
```

El programa de entrada de pedidos ORD040C está en la biblioteca DISTLIB.

Si utiliza mensajes de solicitud durante la entrada del mandato y se le pide un nombre calificado, recibirá mensajes de solicitud que le pedirán el nombre del objeto y de la biblioteca. en la mayoría de mandatos, puede especificar un nombre de biblioteca determinado, especificar *CURLIB (la biblioteca actual del trabajo) o utilizar una lista de bibliotecas. Las listas de bibliotecas se describen en el apartado siguiente.

Subtemas

4.3.1 Listas de Bibliotecas

En los mandatos en los que se puede especificar un nombre calificado, puede omitir la especificación del nombre de biblioteca. Si lo hace así, puede suceder lo siguiente:

- En un mandato de creación, el objeto se crea y se coloca en la biblioteca actual del usuario, *CURLIB, o en una biblioteca del sistema, en función del tipo de objeto. Por ejemplo, los programas se crean y se colocan en *CURLIB; las listas de autorización se crean y se colocan en la biblioteca del sistema, QSYS.

- En el caso de los mandatos que no son de creación, el sistema utiliza normalmente una lista de bibliotecas para localizar el objeto.

Las listas de bibliotecas utilizadas por el sistema AS/400 constan de las cuatro partes siguientes.

Parte del sistema La parte del sistema de la lista de bibliotecas contiene objetos que son necesarios para el sistema.

Bibliotecas productos En la lista de bibliotecas pueden incluirse dos bibliotecas de productos. El sistema utiliza estas bibliotecas para dar soporte a idiomas y programas de utilidad que dependen de bibliotecas distintas de QSYS para procesar sus mandatos.

Los mandatos y menús de usuario también pueden especificar una biblioteca de productos en el parámetro PRDLIB de los mandatos Crear Mandato (CRTCMD) y Crear Menú (CRTMNU) para asegurarse de que se encontrarán los objetos dependientes.

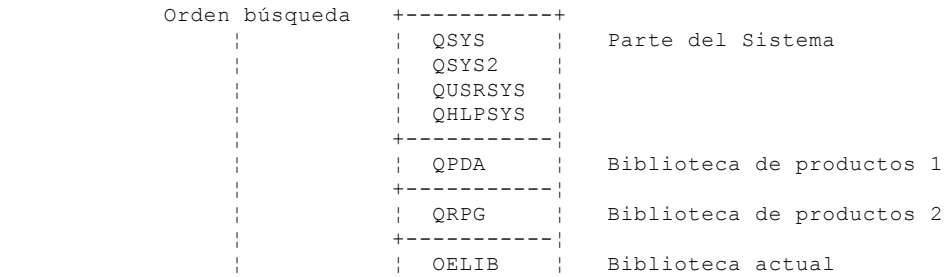
Las bibliotecas de productos están gestionadas por el sistema, que coloca automáticamente las bibliotecas de productos (como QRPG) en una posición de biblioteca de productos reservada dentro de la lista de biblioteca cuando es necesario. Una biblioteca de productos puede ser un duplicado de la biblioteca actual o de una biblioteca en la parte del usuario de la lista de biblioteca.

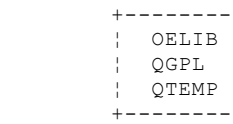
Biblioteca actual La biblioteca actual puede ser, pero no necesariamente, un duplicado de cualquier biblioteca de la lista de bibliotecas. El valor *CURLIB (biblioteca actual) puede utilizarse en la mayoría de los mandatos como un nombre de biblioteca para representar cualquier biblioteca especificada como biblioteca actual para un trabajo determinado. Si no existe biblioteca actual en la lista de bibliotecas, y se especifica *CURLIB como biblioteca, se utiliza QGPL. Puede cambiar la biblioteca actual para un trabajo determinado utilizando el mandato Cambiar Biblioteca Actual (CHGCURLIB) o Cambiar Lista de Bibliotecas (CHGLIBL).

Parte de usuario La parte del usuario de la lista de biblioteca contiene aquellas bibliotecas a las que se remiten los usuarios del sistema y las aplicaciones. La parte del usuario y las bibliotecas actual y de productos puede ser diferentes para cada trabajo en el sistema. Existe un límite de 25 bibliotecas.

Para obtener una lista de las bibliotecas suministradas con el sistema o de las que se pueden instalar opcionalmente, consulte la publicación *Programación - Resumen de consulta*.

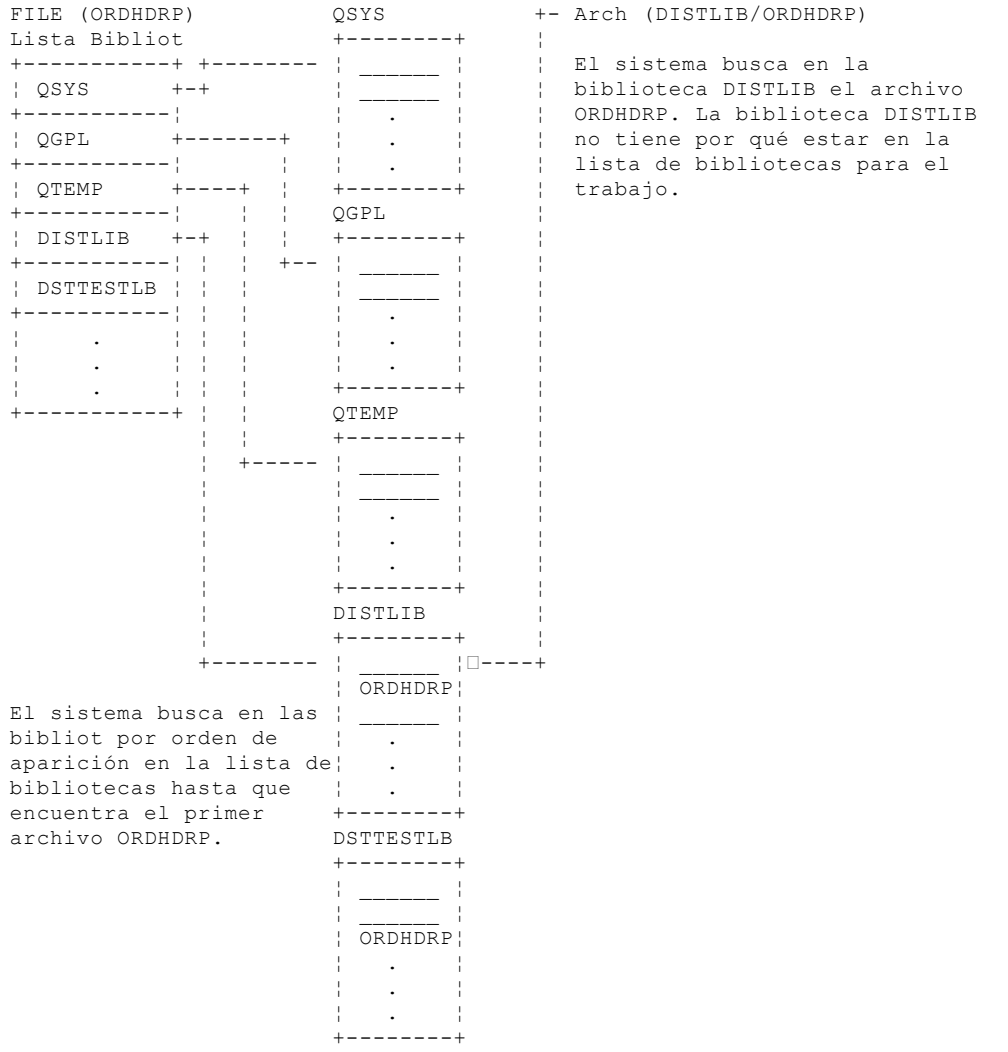
El diagrama siguiente muestra un ejemplo de la estructura de la lista de bibliotecas:





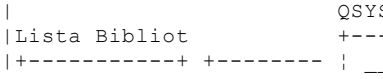
Nota: El sistema coloca la biblioteca QPDA en la biblioteca de productos 1 cuando se utiliza el Programa de Utilidad para Entrada del Fuente(SEU). Cuando se utiliza el SEU para comprobar la sintaxis del código fuente, se puede añadir una segunda biblioteca de productos a la biblioteca de productos 2. Por ejemplo, si se está comprobando la sintaxis del fuente RPG, entonces QPDA es la biblioteca de productos 1 y QRPG es la biblioteca de productos 2. En la mayoría de las demás funciones no se utiliza la biblioteca de productos 2.

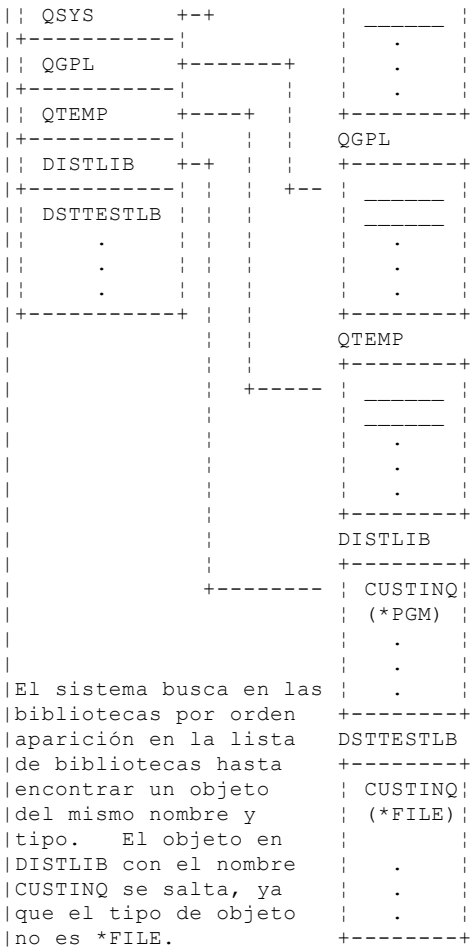
Con el uso de una lista de bibliotecas se simplifica la búsqueda de objetos en el sistema. Cada trabajo está asociado a una lista de bibliotecas. Cuando se utiliza una lista de bibliotecas para encontrar un objeto, se ha de buscar cada biblioteca de la lista por orden de aparición hasta encontrar el objeto con nombre y tipo especificados. Si hay dos o más objetos con el mismo tipo y nombre en la lista, se obtiene el objeto de la biblioteca que aparece primero en la lista de bibliotecas. El diagrama siguiente muestra las búsquedas efectuadas para un objeto cuando se utiliza la lista de bibliotecas (*LIBL) y cuando se especifica un nombre de biblioteca.



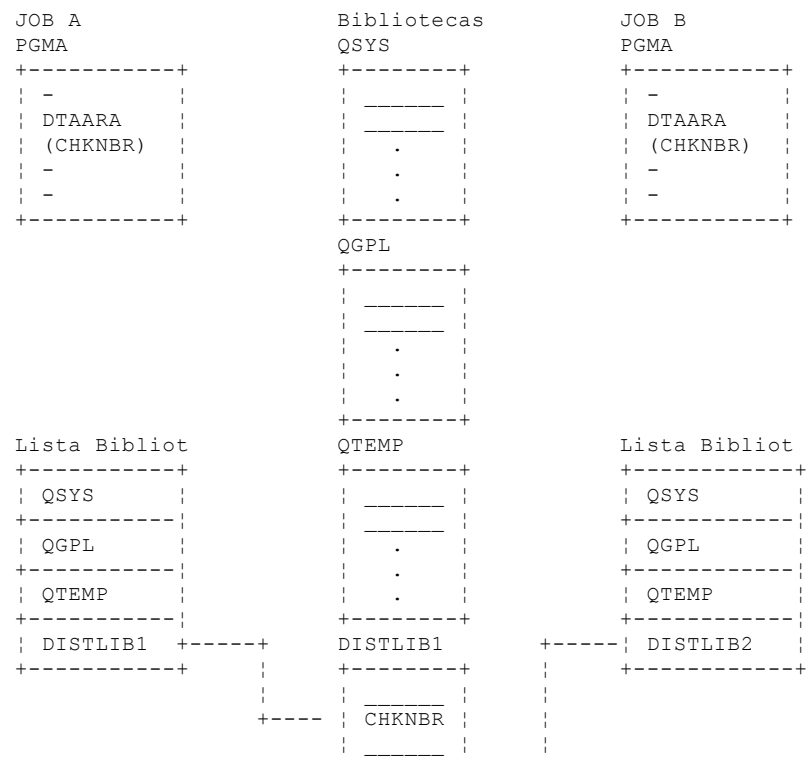
El siguiente diagrama muestra lo que ocurre cuando dos objetos con el mismo nombre pero de diferente tipo se encuentran en la lista de bibliotecas. El sistema buscará el archivo CUSTINQ *FILE en la lista de bibliotecas especificando:

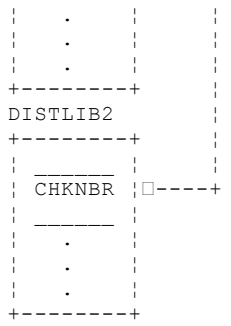
```
DSPOBJD OBJ(*LIBL/CUSTINQ) OBJTYPE(*FILE)
```





Generalmente, una lista de bibliotecas es más flexible y más fácil de utilizar que los nombres calificados. Más importante que la ventaja de no entrar el nombre de la biblioteca es la posibilidad de realizar funciones en una aplicación sobre datos diferentes simplemente utilizando otra lista de bibliotecas sin necesidad de modificar la aplicación. Por ejemplo, el programa CL PGMA actualiza un área de datos CHKNBR. Si no se ha especificado el nombre de la biblioteca, el programa puede actualizar el área de datos denominada CHKNBR en distintas bibliotecas, según la utilización de la lista de bibliotecas. Por ejemplo, suponga que JOBA y JOBB llaman a PGMA, tal como se muestra en la ilustración siguiente:

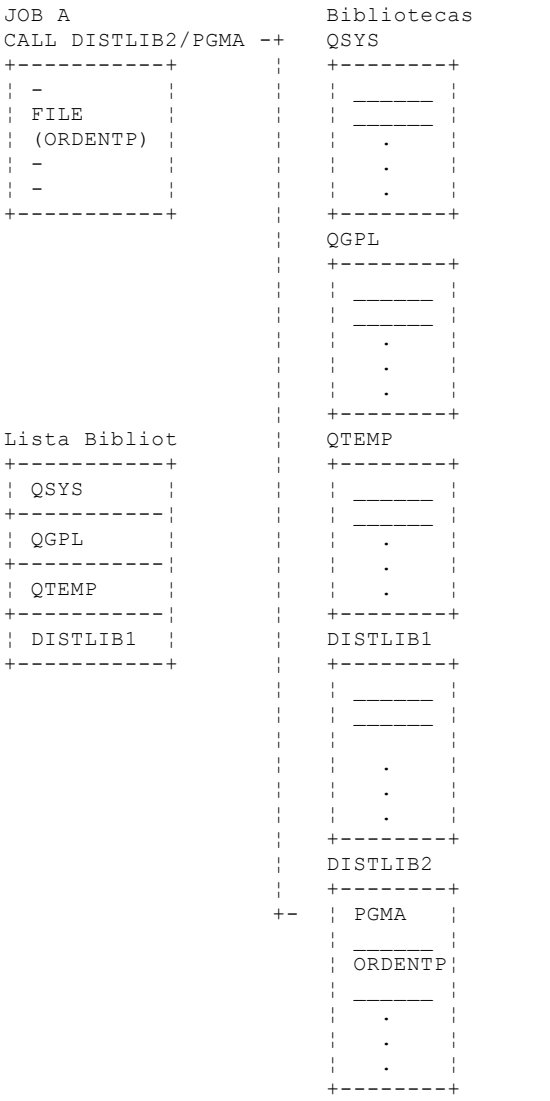




Sin embargo, la utilización de un nombre calificado ofrece ventajas en cualquiera de las situaciones siguientes:

- Cuando el objeto que se está utilizando no está en la lista de bibliotecas para el trabajo
- Cuando hay más de un objeto con el mismo nombre en la lista de bibliotecas y desea uno de una biblioteca específica
- Cuando quiere asegurarse de que se utiliza una biblioteca específica, por razones de seguridad.

Sin embargo, si se llama a un programa utilizando un nombre calificado y el programa intenta abrir archivos cuyos nombres no están calificados, los archivos no se abren si no están en la lista de bibliotecas, tal como se muestra en el ejemplo siguiente:



La llamada a PGMA se realiza con éxito porque el nombre del programa está calificado en el mandato CALL. Sin embargo, cuando el programa intenta abrir el archivo ORDENTP, la operación de apertura falla porque el archivo no se encuentra en ninguna de las bibliotecas de la lista y su nombre no está calificado. Si se añadiese la biblioteca DISTLIB2 a la lista de bibliotecas o se utilizase un nombre de archivo calificado, el programa

podría abrir el archivo. Algunos lenguajes de alto nivel no permiten la especificación de un nombre calificado. Utilizando un mandato Alterar Temporalmente (OVRxxx), se puede especificar un nombre calificado.

Subtemas
4.3.1.1 Una Lista de Bibliotecas del Trabajo
4.3.1.2 Cambio de la Lista de Bibliotecas
4.3.1.3 Consideraciones para la Preparación de una Lista de Bibliotecas

4.3.1.1 Una Lista de Bibliotecas del Trabajo

Cada lista de bibliotecas de un trabajo consta de cuatro partes como máximo: una parte de sistema, una parte de usuario y las bibliotecas actual y de productos. La parte del sistema es la única que *siempre* se incluye en la lista de bibliotecas.

Cuando se envía el sistema, el valor del sistema QSYSLIBL contiene los nombres de las bibliotecas que pasarán a ser la parte del sistema de la lista de bibliotecas. Los valores suministrados son QSYS, QSYS2, QHLPSYS y QUSRSYS. El valor del sistema QUSRLIBL contiene los nombres de las bibliotecas que se convertirán en la parte de usuario de la lista de bibliotecas.

QSYSLIBL puede contener 15 nombres de bibliotecas y QUSRLIBL puede contener 25. Para cambiar la parte de sistema de la lista de bibliotecas de un trabajo, utilice el mandato Cambiar Lista de Bibliotecas del Sistema (CHGSYSLIBL). Para cambiar el valor de QSYSLIBL o de QUSRLIBL, utilice el mandato Cambiar Valor del Sistema (CHGSYSVAL). Un cambio en estos valores del sistema repercute en los trabajos nuevos que se inician posteriormente.

4.3.1.2 Cambio de la Lista de Bibliotecas

Cuando se está ejecutando un trabajo, puede añadir o suprimir entradas de la lista de bibliotecas, para lo cual puede utilizar tanto el mandato Añadir Entrada de Lista de Bibliotecas (ADDLIBLE), como el mandato Suprimir Entrada de Lista de Bibliotecas (RMVLIBLE) o, si lo prefiere, también puede modificar la lista de bibliotecas mediante la utilización de los mandatos CHGLIBL o EDTLIBL. Estos mandatos cambian la parte del usuario de la lista de bibliotecas, pero no la parte del sistema.

La biblioteca actual puede añadirse o cambiarse utilizando el mandato Cambiar Biblioteca Actual (CHGCURLIB) o el mandato CHGLIBL. La biblioteca actual también puede cambiarse en el perfil del usuario, en el inicio de sesión o en el mandato Someter Trabajo (SBMJOB). Las bibliotecas de productos no pueden añadirse utilizando un mandato CL; el sistema añade estas bibliotecas cuando se ejecuta el mandato o menú que las utiliza. Las bibliotecas de producto no pueden cambiarse con un mandato CL; sin embargo, pueden cambiarse con la API Cambiar Lista de Bibliotecas (QLICHGLL).

Cuando se utilizan estos mandatos, el cambio de la lista de bibliotecas afecta sólo al trabajo en el que se ejecuta el mandato, y el cambio tiene efecto sólo mientras se está procesando ese trabajo, o hasta que vuelve a cambiar la lista de biblioteca del trabajo. Cuando la lista de biblioteca se cambia mediante la utilización de estos mandatos, las bibliotecas deben existir cuando se ejecuta este mandato. No se puede suprimir una biblioteca si ésta existe en la lista activa de bibliotecas de usuario.

Cuando un trabajo arranca, la parte de usuario de la lista de bibliotecas está determinada por los valores contenidos en la descripción del trabajo o por valores especificados en el mandato SBMJOB. Puede especificarse un valor de *SYSVAL, lo que hace que las bibliotecas especificadas por el valor del sistema QUSRLIBL se conviertan en la parte de usuario de la lista de bibliotecas. Si se han especificado nombres de bibliotecas en la descripción de trabajo y el mandato Trabajo por Lotes (BCHJOB) o SBMJOB, los nombres de bibliotecas especificados en estos mandato alteran temporalmente las bibliotecas especificadas en la descripción de trabajo y el valor del sistema QUSRLIBL.

A continuación se muestra el orden en que la parte de usuario de la lista de bibliotecas especificada en QUSRLIBL queda alterada temporalmente por mandatos para trabajos individuales:

- En la descripción de trabajo puede especificar una lista de bibliotecas que, cuando se ejecuta el trabajo, altera temporalmente la lista de bibliotecas especificada en QUSRLIBL. (Consulte la publicación *Gestión de Trabajos* para obtener información referente a descripciones de trabajos.)

- Cuando somete un trabajo utilizando el mandato BCHJOB o SBMJOB, puede especificar una lista de bibliotecas en el mandato. Esta lista altera temporalmente la lista de bibliotecas especificada en la descripción de trabajo o en el valor del sistema QUSRLIBL.

- Cuando somete un trabajo utilizando el mandato SBMJOB, puede especificar *CURRENT (el valor por omisión) para la lista de bibliotecas. *CURRENT indica que se utiliza la lista de bibliotecas del trabajo que emite el mandato SBMJOB.

- En un trabajo, puede utilizar ADDLIBLE, RMVLIBLE o CHGLIBL. Estos mandatos alteran cualquier especificación anterior de la lista de bibliotecas.

- La biblioteca actual para el trabajo puede cambiarse utilizando el mandato CHGCURLIB o CHGLIBL.

En lugar de entrar el mandato CHGLIBL cada vez que desee cambiar la lista de bibliotecas, puede situar dicho mandato en un programa CL:

```
  PGM  /* SETLIBL - Establecer lista de bibliotecas */
  CHGLIBL LIBL(APPDEVLIB QGPL QTEMP)
  ENDPGM
```

Si normalmente trabaja con esta lista de bibliotecas, podría preparar un programa inicial para establecer la lista de bibliotecas en lugar de llamar al programa cada vez:

```
  PGM  /* Programa inicial para QPGMR */
  CHGLIBL LIBL(APPDEVLIB QGPL QTEMP)
  TFRCTL PGM(QPGMMENU)
         ENDPGM
```

Es necesario crear este programa cambiar el perfil de usuario al cual se

aplicará para especificar el nuevo programa inicial. El control se transfiere entonces de este programa al programa QPGMMENU, que visualiza el Menú del Programador.

Si necesita ocasionalmente añadir una biblioteca a la lista de bibliotecas especificada en el programa inicial, puede utilizar el mandato ADDLIBLE para añadir la biblioteca a la lista de bibliotecas. Por ejemplo, el mandato siguiente añade la biblioteca JONES al final de la lista de bibliotecas:

```
  ADDLIBLE LIB(JONES) POSITION(*LAST)
```

Si parte de su trabajo requiere una lista de bibliotecas distinta, puede escribir un programa CL que guarde la lista de bibliotecas actual y la restaure posteriormente, como es el caso del siguiente programa.

```
         PGM
         DCL &LIBL *CHAR 275
         DCL &CMD *CHAR 285
  (1)    RTVJOBA USRLIBL(&LIBL)
  (2)    CHGLIBL (QGPL QTEMP)
         .
         .
         .
  (3)    CHGVAR &CMD ('CHGLIBL (' *CAT &LIBL *TCAT ')')
  (4)    CALL QCMDEXC (&CMD 285)
         .
         .
         .
         ENDPGM
```

(1) Mandato para salvar la lista de bibliotecas. La lista de bibliotecas se almacena en la variable &LIBL. Cada nombre de biblioteca ocupa 10 bytes (rellenados con espacios en blanco por la derecha si es necesario) y hay un espacio en blanco entre cada nombre de biblioteca.

(2) Este mandato cambia la lista de bibliotecas tal como requiere la función siguiente.

(3) El mandato Cambiar Variable (CHGVAR) crea un mandato CHGLIBL en la variable &CMD.

(4) Se llama a QCMDEXC para procesar la serie de mandatos en la variable &CMD. El mandato CHGVAR se necesita antes de llamar a QCMEDXC porque no puede realizarse la concatenación en el mandato CALL.

4.3.1.3 Consideraciones para la Preparación de una Lista de Bibliotecas

Al preparar y utilizar una lista de bibliotecas debe tener en cuenta lo siguiente:

- Las bibliotecas en una lista de bibliotecas deben existir en el sistema. Se accede a los valores del sistema QSYSLIBL y QUSRLIBL cuando se arranca el OS/400. Si una biblioteca mencionada en alguno de dichos valores no existe en el sistema, se envía un mensaje a la cola de mensajes del operador del sistema (QSYSOPR), se pasa por alto la biblioteca y el OS/400 se arranca sin la biblioteca. Una vez arrancado el OS/400, no pueden suprimirse bibliotecas de la lista de bibliotecas de ningún trabajo activo. Si cualquier biblioteca de la lista de bibliotecas especificada en la descripción de trabajo o en uno de los mandatos Trabajo (JOB) o Someter Trabajo (SBMJOB) no existe o no está disponible, el trabajo no se arranca.

- Todos los usuarios que necesitan utilizar las bibliotecas de una lista de bibliotecas deben estar autorizados para ello. Para inicializar una lista de bibliotecas (por ejemplo, en un mandato Someter Trabajo [SBMJOB], Trabajo [JOB], o Crear Descripción de Trabajo [CRTJOBD]), un usuario debe tener autorización operativa sobre objetos para las bibliotecas; de lo contrario el trabajo no se arranca. Un usuario debe tener también la autorización *USE para bibliotecas añadidas a la lista de bibliotecas mediante los mandatos Añadir Entrada en Lista de Bibliotecas (ADDLIBLE) o Cambiar Lista de Bibliotecas (CHGLIBL).

- Cuando un programa que se ejecuta bajo un perfil de usuario adoptado añade una biblioteca a la lista de bibliotecas a la que el usuario actual no está autorizado y no la suprime de la lista antes de finalizar, el usuario conserva el acceso (autorización *USE) a la biblioteca una vez concluido el programa. Esto sólo sucede cuando se especifica *LIBL para acceder a los objetos.

- El rendimiento del sistema es mejor cuanto más corta es la lista de bibliotecas.

4.3.2 Visualización de una Lista de Biliotecas

Puede utilizar el mandato Visualizar Lista de Bibliotecas (DSPLIBL) para visualizar la lista de bibliotecas correspondiente a un trabajo que se está ejecutando. La pantalla contiene una lista de todas las bibliotecas de la lista de bibliotecas en el mismo orden en el que aparecen en la lista de bibliotecas.

También puede visualizar la lista de bibliotecas de un trabajo activo utilizando el mandato Visualizar Trabajo (DSPJOB) y seleccionando la opción 13 del menú Visualizar Trabajo.

4.3.3 Utilización de Nombre de Objeto Genéricos

Es posible que en ocasiones desee buscar más de un objeto (aunque sólo haya uno) cuando los nombres de los objetos comiencen con los mismos caracteres. Este tipo de búsqueda se denomina *búsqueda genérica*, y puede utilizarse en varios mandatos.

Si quiere utilizar la búsqueda genérica, especifique en el mandato un nombre genérico en lugar del nombre del objeto. Un nombre genérico consiste en una serie de caracteres comunes a todos los nombres de los objetos que identifica un grupo de objetos y acaba con un * (asterisco). La función solicitada se llevará a cabo en todos los objetos cuyos nombres comiencen con los caracteres especificados y a los que esté autorizado. Por ejemplo, si entra el mandato Visualizar Descripción de Objeto (DSPOBJD) y utiliza el nombre genérico ORD*, aparecerán las descripciones de los objetos que comiencen por ORD.

Una búsqueda genérica puede estar limitada por los siguientes calificadores de biblioteca en el nombre genérico (el valor del parámetro de nombre de biblioteca se da entre paréntesis, si se aplica):

- Una biblioteca especificada. La operación solicitada se realiza en los objetos denominados genéricamente en la biblioteca especificada solamente.

- La lista de bibliotecas para el trabajo (*LIBL). Se busca en las bibliotecas en el orden en el que figuran en la lista de bibliotecas. La operación que se solicitó se realiza en los objetos con nombre genérico que están en las bibliotecas especificadas en la lista de bibliotecas para el trabajo.

- La biblioteca actual para el trabajo (*CURLIB). Se busca en la biblioteca actual para el trabajo. Si no existe ninguna biblioteca actual, se utiliza QGPL.

- Todas las bibliotecas en la parte de usuario de la lista de bibliotecas para el trabajo (*USRLIBL). Se busca en las bibliotecas en el orden en que figuran en la lista de bibliotecas, incluyendo la biblioteca actual (*CURLIB). La operación que se solicitó se realiza en los objetos con nombre genérico que están en las bibliotecas especificadas en la parte de usuario de la lista de bibliotecas para el trabajo.

- Se busca en todas las bibliotecas de usuario para las que tiene autorización (*ALLUSR) y las siguientes bibliotecas que empiezan por la letra Q: QDSNX, QGPL, QGPL38, QPFRDATA, QRCL, QS36F, QUSER38, QUSRINFSKR, QUSRSYS y QUSRVxRxMx (VxRxMx es la versión, release, y nivel de modificación de la biblioteca). Se busca en las bibliotecas en orden alfanumérico. No se busca en las siguientes bibliotecas del entorno S/36 que empiezan por # cuando se ha especificado *ALLUSR: #CGULIB, #COBLIB, #DFULIB, #DSULIB, #RPGLIB, #SDALIB y #SEULIB. La operación que solicitó se efectúa en los objetos con nombre genérico que están en todas las bibliotecas de usuario a las que está autorizado.

- Todas las bibliotecas del sistema para las que está autorizado (*ALL). Se busca en las bibliotecas en orden alfanumérico. La operación que solicitó se realiza en los objetos con nombre genérico que están en todas las bibliotecas del sistema a las que está autorizado.

Para obtener información referente a operaciones utilizando funciones genéricas, consulte el manual CL Reference.

4.3.4 Búsqueda de Múltiples Objetos o de Un Solo Objeto

En todos los mandatos en los que puede especificar un nombre genérico, puede especificar un nombre de objeto (sin asterisco), así como buscar varios objetos. Si especifica el nombre de un objeto junto con *ALL o *ALLUSR para el nombre de la biblioteca, el sistema busca diversos objetos, y la búsqueda devuelve los objetos con el nombre y tipo indicados para los cuales está autorizado. Si especifica un nombre genérico, *ALL, *ALLUSR, o una biblioteca con el nombre de un objeto, puede especificar todos los tipos de objetos soportados (o tipos de objeto *ALL).

4.4 Utilización de Bibliotecas

Una biblioteca es un objeto que se utiliza para agrupar objetos relacionados, así como para buscar objetos mediante el nombre. Por lo tanto, una biblioteca es un directorio de un grupo de objetos.

Puede utilizar las bibliotecas para:

- Agrupar ciertos objetos para usuarios individuales. Esto ayuda a gestionar los objetos en el sistema. Por ejemplo, puede colocar todos los archivos que un usuario JOE puede utilizar en una biblioteca JOELIB.

- Agrupar todos los objetos utilizados para una aplicación individual. Por ejemplo, puede colocar todos los programas y archivos de entrada de pedidos en una biblioteca de entrada de pedidos DISTLIB. Sólo se precisa añadir una biblioteca a la lista de bibliotecas para asegurarse de que todos los programas y archivos de entrada de pedidos están en la lista. Esto le ofrece ventajas si no desea especificar un nombre de biblioteca cada vez que utiliza un programa o archivo de entrada de pedidos.

- Garantizar la seguridad. Por ejemplo, puede especificar qué usuarios tienen autorización para utilizar la biblioteca y lo que están autorizados a hacer con ella.

- Simplificar la seguridad mediante una lista automática de autorizaciones y una asignación de autorización de uso público para aquellos objetos creados recientemente según el valor del parámetro CRTAUT de la biblioteca. Los atributos de auditoría para objetos recién creados, pueden establecerse basándose en el valor de parámetro
| Crear Auditoría de Objeto (CRTOBJAUD).

- Simplificar las operaciones de salvar/restaurar agrupando objetos que se salvan y restauran al mismo tiempo en la misma biblioteca. Puede utilizar el mandato Salvar Biblioteca (SAVLIB) en lugar de salvar objetos individualmente utilizando el mandato Salvar Objeto (SAVOBJ).

- Utilizar diversas bibliotecas para pruebas. Consulte Apéndice A para obtener más información.

- Utilizar varias bibliotecas de producción. Por ejemplo, puede utilizar una biblioteca de producción para archivos fuente y para la creación de objetos, una para los archivos y programas de aplicación, una para los objetos que se salvan con poca frecuencia y una para los objetos que se salvan frecuentemente.

Con varias bibliotecas resulta más sencillo utilizar los objetos. Por ejemplo, se pueden tener dos archivos con el mismo nombre pero en bibliotecas diferentes de modo que puede utilizar una de prueba y la otra para el proceso normal. Mientras no se especifique el nombre de biblioteca en el programa, no es necesario cambiar el nombre de archivo en el programa para el proceso de prueba ni para el proceso normal. Puede controlar qué biblioteca utiliza empleando la lista de bibliotecas. (Los objetos del mismo tipo pueden tener el mismo nombre solamente si están en bibliotecas distintas.)

Existen dos tipos de bibliotecas: de producción y de prueba. Las bibliotecas de producción se utilizan para el proceso normal. En la modalidad de depuración, puede evitar que se actualicen los archivos de base de datos de las bibliotecas de producción. Mientras se trabaja en la modalidad de depuración, los archivos de las bibliotecas de prueba se pueden actualizar sin especificaciones exclusivas. (Consulte la sección Apéndice A para obtener más información acerca de la utilización de bibliotecas de prueba.)

Subtemas

4.4.1 Creación de una Biblioteca

Para crear una biblioteca, utilice el mandato Crear Biblioteca (CRTLIB). Por ejemplo, el mandato CRTLIB siguiente crea una biblioteca que se utilizará para contener programas y archivos de entrada de pedidos. Se trata de una biblioteca de producción y se denomina DISTLIB. La autorización que se otorga por omisión al público evita que un usuario tenga acceso a la biblioteca. A cualquier objeto que se cree dentro de la biblioteca se le dará la autorización de uso público por omisión *CHANGE basándose en el valor CRTAUT.

```
  CRTLIB   LIB(DISTLIB) TYPE(*PROD) CRTAUT(*CHANGE) CRTOBJAUD(*USRPRF) +
           AUT(*EXCLUDE) TEXT('Biblioteca de distribución')
```

No debe crear ninguna biblioteca con un nombre que empiece por la letra Q. Durante una búsqueda genérica, el sistema supone que la mayoría de las
|bibliotecas cuyos nombres empiezan con la letra Q (como QRPG o QPDA) son
|bibliotecas del sistema. Consulte "Utilización de Nombre de Objeto Genéricos" en el tema 4.3.3 para obtener mas información.

4.4.2 Especificación de Autorización para Bibliotecas

A continuación se describen las autorizaciones que pueden otorgarse a los usuario para las bibliotecas. Consulte el manual Security - Reference para obtener mas información.

Subtemas
4.4.2.1 Autorización sobre Objeto
4.4.2.2 Autorización sobre Datos
4.4.2.3 Autorizaciones combinadas

|*4.4.2.1 Autorización sobre Objeto*

La **autorización operativa sobre objetos** para una biblioteca otorga autorización al usuario para visualizar la descripción de una biblioteca.

La **autorización de gestión de objetos** para una biblioteca permite:

- Otorgar y revocar autorizaciones. Sólo puede otorgar y revocar las autorizaciones que posea. Solamente el propietario del objeto o un usuario con autorización *ALLOBJ pueden otorgar autorización de gestión de objetos para una biblioteca.
- Cambiar el nombre de la biblioteca.

|La **autorización de existencia de objetos** y la **autorización de utilización** |proporcionan al usuario autorización para suprimir una biblioteca.

|La **autorización de existencia de objetos** y la **autorización de operatividad** |**de objetos** proporcionan al usuario autoridad para transferir la propiedad |de una biblioteca.

4.4.2.2 Autorización sobre Datos

La **autorización de adición** y la **autorización de lectura** para una bilioteca, permiten al usuario crear un objeto nuevo en la biblioteca o trasladar un objeto a la biblioteca.

La **autorización de actualización** y la **autorización de ejecución** para una biblioteca permiten a un usuario cambiar el nombre de un objeto de la biblioteca, siempre y cuando tenga autorización sobre dicho objeto.

La **autorización de supresión** permite al usuario eliminar entradas de un objeto. La autorización de supresión para una biblioteca no permite al usuario eliminar objetos de la biblioteca. La autorización para el objeto de la biblioteca se utiliza para determinar si el objeto se puede suprimir.

La **autorización de ejecución** permite al usuario buscar un objeto en la biblioteca.

|4.4.2.3 Autorizaciones combinadas

La **autorización *USE*** para una biblioteca (que consta de autorización de operación sobre objeto, autorización de lectura y autorización de ejecución) incluye autorización para:

- Utilizar una biblioteca para buscar un objeto
- Visualizar el contenido de una biblioteca
- Colocar una biblioteca en la lista de bibliotecas
- Salvar una biblioteca (si tiene suficiente autorización sobre el objeto)
- Suprimir objetos de la biblioteca (si el usuario tiene autorización para los objetos de la biblioteca)

La **autorización *CHANGE*** para una biblioteca (que consta de autorización de operación sobre objeto y autorización sobre todos los datos para la biblioteca) proporciona autorización para:

- Utilizar una biblioteca para buscar un objeto
- Visualizar el contenido de una biblioteca
- Colocar una biblioteca en la lista de bibliotecas
- Salvar una biblioteca (si tiene suficiente autorización sobre el objeto)
- Suprimir objetos de la biblioteca (si el usuario tiene autorización para los objetos de la biblioteca)
- Añadir objetos a la biblioteca.

La **autorización *ALL*** proporciona todas las autorizaciones sobre objetos y sobre datos. El usuario puede suprimir la biblioteca, especificar su seguridad, cambiar la biblioteca, y visualizar la descripción y el contenido de la misma.

La **autorización *EXCLUDE*** impide que los usuarios accedan a un objeto.

Para visualizar la autorización asociada con su biblioteca, puede utilizar el mandato Visualizar Autorización sobre Objeto (DSPOBJAUT).

4.4.3 Consideraciones de Seguridad para Objetos

Cuando el sistema accede a un objeto al que el usuario hace referencia, éste determina si el usuario está autorizado a utilizar el objeto y a usarlo en la forma que se solicita. Generalmente, debe estar autorizado a dos niveles:

- Debe estar autorizado a utilizar el objeto en el que se ha solicitado la realización de una función.
- Debe estar autorizado a la biblioteca que contiene el objeto. Si utiliza una lista de bibliotecas, debe estar autorizado a las bibliotecas de la lista.

La autorización sobre objetos está controlada por las funciones de seguridad del sistema, que incluyen lo siguiente:

- El propietario del objeto y los usuarios con autorización especial *ALLOBJ poseen toda la autorización para un objeto y pueden otorgar y revocar la autorización a otros usuarios.

- Los usuarios tienen autorización de uso público cuando no se ha otorgado autorización privada sobre un objeto.

El manual Security - Reference explica detalladamente los tipos de autorización que pueden otorgarse para un objeto y qué autorización necesita un usuario para ejecutar una función sobre ese objeto. La autorización que puede otorgarse para las bibliotecas se trata en el apartado "Especificación de Autorización para Bibliotecas" en el tema 4.4.2.

Se aplicarán consideraciones especiales al escribir un programa que deba estar asegurado (por ejemplo, un programa que adopta el perfil de usuario del responsable de seguridad). Consulte el manual Security - Reference para obtener información referente a cómo escribir estos programas.

4.4.4 Autorización de Uso Público por Omisión para Objetos de Reciente Creación

Cuando se crean objetos en una biblioteca, la autorización de uso público
|para el objeto se establecerá, por omisión, mediante el valor CRTAUT de la
|biblioteca.

|Mediante la especificación de:

```
  CRTLIB LIB(TESTLIB) CRTAUT(*USE) AUT(*LIBCRTAUT)
```

Se crea la biblioteca TESTLIB. Todos los objetos creados en la biblioteca TESTLIB tendrán, por omisión, la autorización de uso público de *USE. La autorización de uso público para la biblioteca TESTLIB viene determinada por el valor CRTAUT de la biblioteca QSYS.

Mediante la especificación de:

```
  CRTDTAARA DTAARA(TESTLIB/DTA1) TYPE(*CHAR) +
            AUT(*LIBCRTAUT)

  CRTDTAARA DTAARA(TESTLIB/DTA2) TYPE(*CHAR) +
            AUT(*EXCLUDE)
```

El área de datos DTA1 se crea en la biblioteca TESTLIB. La autorización de uso público de DTA1 es *USE basado en el valor CRTAUT de la biblioteca TESTLIB.

El área de datos DTA2 se crea en la biblioteca TESTLIB. La autorización de uso público de DTA2 es *EXCLUDE. *EXCLUDE se especificó en el parámetro AUT del mandato Crear Área de Datos (CRTDTAARA).

También se puede utilizar una lista de autorizaciones para asegurar un objeto cuando se crea en de una biblioteca. Mediante la especificación de:

```
  CRTAUTL AUTL(PAYROLL)
  CRTLIB  LIB(PAYLIB) CRTAUT(PAYROLL) +
          AUT(*EXCLUDE)
```

Se crea una lista de autorizaciones llamada PAYROLL. La biblioteca PAYLIB se crea con una autorización de uso público de *EXCLUDE. Por omisión, un objeto creado en la biblioteca PAYLIB se asegura mediante la lista de autorizaciones PAYROLL.

Mediante la especificación de:

```
  CRTPF FILE(PAYLIB/PAYFILE) +
        AUT(*LIBCRTAUT)

  CRTPF FILE(PAYLIB/PAYACC) +
        AUT(*CHANGE)
```

El archivo PAYFILE se crea en la biblioteca PAYLIB. El archivo PAYFILE se asegura con la lista de autorizaciones PAYROLL. La autorización de uso público del archivo PAYFILE queda establecida en *AUTL como parte del mandato Crear Archivo Físico (CRTPF). *AUTL indica que la autorización de uso público para el archivo PAYFILE se toma de la lista de autorizaciones que asegura el archivo PAYFILE, que es la lista de autorizaciones PAYROLL.

El archivo PAYACC se crea en la biblioteca PAYLIB. La autorización de uso público para el archivo PAYACC es *CHANGE puesto que así está especificado en el parámetro AUT del mandato CRTPF.

Nota: El valor *LIBCRTAUT del parámetro AUT que existe en la mayoría de los mandatos CRT indica que la autorización de uso público para el objeto está fijada en el valor CRTAUT de la biblioteca en la que se está creando el objeto.

El valor CRTAUT de la biblioteca especifica la autorización por omisión para uso público de los objetos creados en la biblioteca. Estos posibles valores son:

***SYSVAL**	La autorización de uso público para los objetos que se crea es el valor especificado en el valor del sistema QCRTAUT
***ALL**	Todas las autorizaciones de uso público
***CHANGE**	Autorización de cambio
***USE**	Autorización de uso
***EXCLUDE**	Autorización de exclusión

nombre de lista de autorizaciones
La lista de autorizaciones asegura el objeto

4.4.5 Atributo de Auditoría Por Omisión para Objetos Recién Creados

Cuando se crean objetos en una biblioteca, el atributo de auditoría del objeto se establecerá, por omisión, utilizando el valor CRTOBJAUT de la biblioteca. Mediante la especificación de:

```
CRTLIB LIB(PAYROLL) AUT(*EXCLUDE) CRTAUT(*EXCLUDE) CRTOBJAUD(*ALL)
```

se realiza una auditoría sobre todos los objetos creados en la biblioteca PAYROLL para el acceso de lectura y de cambio. Consulte el manual Security - Reference para ver los detalles de la auditoría.

4.4.6 Colocación de Objetos en Bibliotecas

Cuando se crea un objeto, éste se colocará en una biblioteca. Si no especifica una biblioteca, el objeto se coloca en la biblioteca actual del trabajo (*CURLIB) o, en caso de no existir ésta, en QGPL. Cuando se crea una biblioteca, puede especificar la autorización de uso público para objetos creados en la biblioteca, utilzando el parámetro CRTAUT en el mandaro Crear Biblioteca (CRTLIB). Todos los objetos situados en esta biblioteca tomarán la autorización de uso público del valor CRTAUT de la biblioteca. Para especificar una biblioteca, especifique un nombre calificado, es decir, un nombre de biblioteca y un nombre de objeto. Por ejemplo, el siguiente mandato Crear Archivo Físico (CRTPF) crea un archivo físico de entrada de órdenes ORDHDRP que se colocará en DISTLIB.

```
  CRTPF FILE(DISTLIB/ORDHDRP)
```

Para colocar un objeto en una biblioteca, debe poseer autorizaciones de lectura y adición para la biblioteca.

Más de un objeto del mismo tipo no pueden tener el mismo nombre y estar en la misma biblioteca. Por ejemplo, en la biblioteca DISTLIB no puede haber dos archivos con el nombre ORDHDRP. Si se intenta colocar un objeto en una biblioteca que tenga el mismo nombre y tipo que otro que ya estaba en la biblioteca, el sistema rechaza la petición y le envía un mensaje indicando el motivo.

Nota: Utilice la biblioteca QSYS para objetos del sistema solamente. Otros programas bajo licencia no deben restaurarse a la biblioteca, ya que los cambios se pierden.

4.4.7 Supresión y Borrado de Bibliotecas

Cuando se suprime una biblioteca con el mandato Suprimir Biblioteca (DLTLIB), se suprimen tanto los objetos de la biblioteca como la propia biblioteca. Cuando se borra una biblioteca con el mandato Borrar Biblioteca (CLRLIB), se suprimen los objetos de la biblioteca sin eliminar ésta. Para suprimir o borrar una biblioteca, lo único que es preciso especificar es el nombre de la misma. Por ejemplo:

```
DLTLIB LIB(DISTLIB)
```

o:

```
CLRLIB LIB(DISTLIB)
```

Para suprimir una biblioteca, debe poseer autorización de existencia de objetos para la biblioteca y los objetos de ésta, así como autorización de uso para la biblioteca. Si intenta suprimir una biblioteca pero no dispone de autorización de existencia para todos los objetos de la biblioteca, no se suprimen la biblioteca ni los objetos a los que no tiene autorización. Se suprimen todos los objetos para los que sí tiene autorización. Si intenta suprimir una biblioteca, pero no tiene autorización de existencia de objetos para ella, no se suprimirá la biblioteca ni ninguno de los objetos de la misma. Si desea suprimir un objeto determinado (para el que dispone de autorización de existencia de objetos), puede utilizar un mandato de supresión para ese tipo de objeto, como el mandato Suprimir Programa (DLTPGM).

No puede suprimir una biblioteca de una lista de bibliotecas de un trabajo activo. Para suprimirla, debe esperar a que finalice el trabajo. Por ello, debe suprimir la biblioteca antes de que comience el siguiente paso de direccionamiento. Cuando suprime una biblioteca, debe asegurarse de que nadie más necesita la biblioteca ni los objetos de la misma.

Si una biblioteca forma parte de la lista de bibliotecas inicial definida por los valores del sistema QSYSLIBL y QUSRLIBL, es preciso seguir los pasos siguientes para suprimirla:

1. Utilice el mandato Cambiar Valor del Sistema (CHGSYSVAL) para eliminar la biblioteca del valor del sistema que la contiene. (El valor del sistema cambiado no afecta a la lista de bibliotecas de los trabajos que están en ejecución).

2. Utilice el mandato Cambiar Lista de Bibliotecas (CHGLIBL) para cambiar la lista de bibliotecas del trabajo.

   Puede utilizar también los mandatos Cambiar Lista de Bibliotecas del Sistema (CHGSYSLIBL), Añadir Entrada en Lista de Bibliotecas (ADDLIBLE) y Suprimir Entrada de Lista de Bibliotecas (RMVLIBLE) para cambiar la lista de bibliotecas.

3. Utilice el mandato DLTLIB para suprimir la biblioteca y los objetos de la misma.

Nota: La biblioteca QSYS no se puede suprimir y no debe suprimir los objetos que se encuentran en ella. Podría provocar que finalizara el sistema, ya que éste necesita los objetos que están en QSYS para funcionar correctamente. Tampoco debería suprimir la biblioteca QGPL, porque contiene igualmente algunos objetos que le son necesarios al sistema para rendir con efectividad. Asimismo, no debe utilizar la biblioteca QRECOVERY porque está pensada para un uso exclusivo por parte del sistema. Esta biblioteca contiene objetos que necesita el sistema para su correcto funcionamiento.

Si tiene dudas sobre la supresión de objetos que no sean bibliotecas, véase el apartado "Supresión de Objetos" en el tema 4.15.

Para borrar una biblioteca, debe disponer de autorización de existencia sobre la biblioteca y los objetos de la biblioteca y autorización de uso sobre la biblioteca. Si intenta borrar una biblioteca, pero no posee autorización de existencia de objetos para todos los objetos de la biblioteca, no se suprimen los objetos para los que no tiene autorización. Si un objeto está asignado a otro usuario, no se suprime.

4.4.8 Visualización de Nombres de Biblioteca y su Contenido

Puede utilizar el mandato Visualizar Biblioteca (DSPLIB) o Trabajar con Bibliotecas (WRKLIB) para visualizar o imprimir todas las bibliotecas para las cuales tenga autorización, así como para encontrar información básica sobre cada objeto dentro de las bibliotecas.

La información de objetos incluye:

- El nombre y tipo del objeto
- Los atributos del objeto
- El tamaño del objeto
- La descripción entrada para el objeto cuando éste se creó

En el mandato DSPLIB, puede especificar también uno o varios nombres de bibliotecas, en cuyo caso se eludirá la pantalla de selección de bibliotecas. En esta lista, los objetos se agrupan por biblioteca; dentro de cada biblioteca, se agrupan por el tipo de objeto; dentro de cada tipo están listados por orden alfanumérico. El orden de las bibliotecas es uno de los siguientes:

- Si las bibliotecas se especifican en el mandato DSPLIB, se visualizan en el orden en el que están especificadas en el mandato de visualización.
- Si se especifica *LIBL o *USRLIBL en el mandato DSPLIB, el orden de las bibliotecas coincide con el orden que siguen en la lista de bibliotecas para el trabajo.
- Si especifica *ALL o *ALLUSR en el mandato DSPLIB, el orden de las
| bibliotecas es alfanumérico. El usuario debe tener autorización de
| lectura para la biblioteca a visualizar.

Por ejemplo, el mandato DSPLIB siguiente visualiza una lista de los objetos contenidos en DISTLIB:

```
  DSPLIB  LIB(DISTLIB)  OUTPUT(*)
```

El asterisco (*) del parámetro OUTPUT significa que las bibliotecas se mostrarán en la estación de pantalla si se está llevando a cabo un proceso interactivo y se imprimirán si se trata de un proceso por lotes. Para imprimir una lista estando en proceso interactivo, especifique *PRINT en lugar de tomar el valor por omisión *.

Consulte el manual CL Reference para obtener más información y pantallas ejemplo para el mandato DSPLIB.

4.4.9 Visualización y Recuperación de Descripciones de Biblioteca

Puede utilizar los mandatos Visualizar Descripción de Biblioteca (DSPLIBD) y Recuperar Descripción de Biblioteca (RTVLIBD) para visualizar y recuperar la descripción de las bibliotecas.

La información de descripción de la biblioteca incluye:

- Tipo de biblioteca (PROD o TEST)
- Agrupación de almacenamiento auxiliar de la biblioteca
- Autorización de creación de la biblioteca
- Crear auditoría de objeto de la biblioteca
- Texto descriptivo de la biblioteca

4.5 Soporte de Idiomas Nacionales OS/400

El programa bajo licencia OS/400 da soporte a diferentes idiomas nacionales en el mismo sistema. Esto permite que una información pueda presentarse en el idioma nacional del usuario al mismo tiempo que a otro usuario se le presenta información en otro idioma.

El idioma que se utiliza ofrecer la información al usuario (pantallas, mensajes, salida impresa y la información de ayuda en línea) está controlado por la lista de bibliotecas del trabajo. La información puede presentarse en diferentes versiones de idiomas nacionales añadiendo simplemente una biblioteca de idioma nacional a la parte de sistema de la lista de bibliotecas. Para el idioma primario, una **versión de idioma nacional** es el código de ejecución y los datos de texto para cada programa bajo licencia entrado. Para el idioma secundario, son los datos de texto para todos los programas bajo licencia.

La información sobre idiomas para el idioma primario del sistema se almacena en las mismas bibliotecas que los programas para los programas bajo licencia de IBM. Por ejemplo, si el idioma nacional primario del sistema es el inglés, las biblioteca como QSYS, QHLPSYS y QSSP contienen información en inglés. Las bibliotecas QSYS y QHLPSYS se encuentran en la parte del sistema de la lista de bibliotecas. Las bibliotecas para otros
|programas bajo licencia (como QRPGLE para ILE RPG para OS/400*) las añade el sistema a la lista de bibliotecas cuando se necesitan.

Las versiones de idiomas nacionales diferentes del idioma primario del sistema se instalan en bibliotecas de idiomas nacionales secundarios. Cada biblioteca de idioma secundario contiene una sola versión de idioma nacional de las pantallas, mensajes, mandatos y mensajes de solicitud y ayuda para *todos* los programas bajo licencia de IBM. El nombre de una biblioteca de idioma secundario sigue el formato **QSYSnnnn**, siendo **nnnn** un código de característica del idioma. Por ejemplo, el código de característica para el español es 2931; así pues el nombre de la biblioteca del idioma nacional secundario para el español será QSYS2931.

Si un usuario desea que la información aparezca en el idioma nacional primario del sistema, no se necesita ninguna acción especial. Para presentar información en un idioma nacional diferente del idioma nacional primario del sistema, el usuario puede cambiar la lista de bibliotecas para que la biblioteca del idioma nacional que desea aparezca delante de las demás bibliotecas de la lista de bibliotecas que contiene la información sobre idiomas nacionales. Puede utilizar cualquiera de las siguientes opciones para colocar primero la biblioteca del idioma nacional deseado:

|□ Puede utilizar el parámetro SYSLIBLE en CRTSBSD o CHGSBSD para presentar pantallas, mensajes, etcétera, para un idioma específico. Por ejemplo:

```
CRTSBSD SBSD(QSBSD 2928) POOLS((1 *NOTSG)) SYSLIBLE(QSYS2928)
```

□ Puede utilizar el parámetro LIB del mandato CHGSYSLIBL para especificar la biblioteca de idioma nacional que desea al principio de la lista de bibliotecas. Por ejemplo:

```
CHGSYSLIBL  LIB(QSYS2931)
```

□ Puede preparar un programa inicial en el perfil de usuario para especificar la biblioteca de idioma nacional que desea al principio de la lista de bibliotecas para un trabajo interactivo. Esta es una opción muy adecuada si el usuario no desea ejecutar el mandato CHGSYSLIBL en cada inicio de sesión. El programa inicial utiliza el mandato Cambiar Lista de Bibliotecas del Sistema (CHGSYSLIBL) para añadir la biblioteca deseada de idioma nacional al principio de la lista de bibliotecas.

Nota: La autorización otorgada con el mandato CHGSYSLIBL no permite a todos los usuarios ejecutar el mandato.

Para permitir que un usuario ejecute el mandato CHGSYSLIBL sin otorgar al usuario derechos sobre el mandato, puede escribir un programa CL que contenga el mandato CHGSYSLIBL. El responsable de seguridad posee el programa, el cual adopta la autorización del responsable de seguridad cuando se crea. Cualquier usuario con autorización para ejecutar el programa puede utilizarlo para cambiar la parte del sistema de la lista de bibliotecas en el trabajo del usuario. A continuación se facilita un ejemplo de un programa para establecer la lista de bibliotecas para un usuario español.

```
PGM
  CHGSYSLIBL  LIB(QSYS2931)  /* Utilizar información en español */
ENDPGM
```

4.6 Descripción de Objetos

Siempre que utilice un mandato de creación para crear un objeto, puede describir el objeto en un campo de 50 caracteres en el parámetro TEXT del mandato de creación. Algunos mandatos permiten el valor por omisión *SRCMBRTXT, lo cual indica que el texto para el objeto que se crea ha de tomarse del texto del miembro de fuente desde el que se crea dicho objeto. Esto es válido únicamente para objetos creados desde un fuente en archivos fuente de base de datos.

Si la entrada fuente para el mandato de creación es un dispositivo o un archivo incorporado, o si el fuente no se utiliza, el valor por omisión son espacios en blanco. Este texto pasa a formar parte de la descripción del objeto y se puede visualizar mediante el mandato Visualizar Descripción de Objeto (DSPOBJD) o Visualizar Biblioteca (DSPLIB). El texto se puede modificar utilizando el mandato Cambiar Descripción de Objeto (CHGOBJD) o muchos de los mandatos Cambiar (CHGxxx) que son específicos a cada tipo de objeto.

4.7 Visualización de Descripciones de Objeto

Puede utilizar el mandato Visualizar Descripción de Objeto (DSPOBJD) o Trabajar con Objetos (WRKOBJ) para visualizar las descripciones de los objetos. Estas descripciones resultan útiles a la hora de determinar si existen objetos en el sistema pero no se están utilizando. Si utiliza el proceso por lotes, las descripciones se pueden imprimir o grabar en un archivo de base de datos. Si utiliza el proceso interactivo, las descripciones se pueden visualizar, imprimir o grabar en un archivo de base de datos.

Puede visualizar atributos básicos, completos o de servicio para las descripciones de objetos. Dichas descripciones figuran en la tabla siguiente:

Tabla 4-2. Atributos visualizados para descripciones de objetos

Atributos básicos	**Atributos completos**	**Atributos de servicio (véase las Notas)**
□ Nombre de objeto	□ Nombre de objeto	□ Nombre de objeto
□ Nombre de biblioteca	□ Nombre de biblioteca	□ Nombre de biblioteca
□ Tipo de objeto	□ Tipo de objeto	□ Tipo de objeto
□ Atributos ampliados	□ Propietario	□ Archivo fuente y biblioteca
□ Tamaño del objeto	□ Grupo primario	□ Nombre de miembro
□ Descripción del texto (parcial)	□ Atributos ampliados	□ Atributos ampliados
	□ Atributo definido por el usuario	□ Atributo definido por el usuario
	□ Descripción del texto	□ Estado liberado
	□ Fecha y hora de creación	□ Tamaño del objeto
	□ Usuario que creó el objeto	□ Fecha y hora de creación
	□ Sistema en que se creó el objeto	□ Fecha y hora en que el miembro del archivo fuente se actualizó por última vez
	□ Dominio del objeto	□ Nivel del sistema
	□ Fecha y hora del cambio	□ Compilador
	□ Si se han recogido o no datos de utilización	□ Nivel de control de objeto
	□ Fecha de la última utilización	□ Cambiado por el programa
	□ Cuenta de días que se ha utilizado el objeto	□ Cambiado o no por el usuario
	□ Restauración de la cuenta de fechas utilizadas	□ Programa bajo licencia
	□ Permitir cambios por parte del programa	□ Número PTF
	□ Valor de auditoría de objeto	□ ID de APAR
	□ Tamaño del objeto	□ Descripción de texto del objeto o condiciones del estado del objeto
	□ Tamaño fuera de línea	
	□ Estado liberado	
	□ Estado de compresión	
	□ Agrupación de almacenamiento auxiliar	
	□ Objeto desbordado	
	□ Fecha y hora de la operación de salvar	
	□ Fecha y hora de la operación de restaurar	
	□ Mandato salvar	
	□ Tipo de dispositivo	

Notas:

1. El personal de soporte de programación utiliza la información de servicio para determinar el nivel del sistema en el que se creó el objeto y si este objeto se ha modificado desde que se envió. Parte de

esta información puede serle útil porque indica el miembro fuente utilizado para crear un objeto y la última fecha de cambio efectuado en dicho fuente a partir del cual se creó el objeto.

2. Los objetos de biblioteca contienen solamente los *nombres* de los objetos incluidos en la biblioteca. Si utiliza DSPOBJD para el tipo de objeto *LIB, la información del tamaño del objeto hace referencia al tamaño del objeto de biblioteca solamente, no al tamaño total de los objetos incluidos en la biblioteca.

   Puede utilizar la API Recuperar Descripción de Biblioteca (QLIRLIBD) o el mandato DSPLIB OUTPUT(*PRINT) para buscar el tamaño total de la biblioteca.

Utilizando el mandato DSPOBJD o WRKOBJ, puede listar los objetos de una biblioteca a la que tiene autorización, por:

- Nombre
- Nombre genérico
- Tipo
- Nombre o nombre genérico dentro de cada tipo de objeto

Los objetos se listan por biblioteca; dentro de una biblioteca se listan por tipo. Dentro del tipo de objeto, se listan por orden alfanumérico.

Es posible que quiera utilizar el mandato DSPOBJD en un trabajo de proceso por lotes si desea visualizar muchos objetos con la opción *FULL o *SERVICE. La salida puede ir al archivo en spool de impresora e imprimirse, en lugar de aparecer en la estación de pantalla, o bien puede ir al archivo de base de datos. Si dirige la salida a un archivo de base de datos, todos los atributos del objeto se graban en el archivo. Utilice el mandato Visualizar Archivo Descripción de Campo (DSPFFD) para el archivo QADSPOBJ, en la biblioteca QSYS, para ver el formato de registro para este archivo.

El siguiente mandato visualiza las descripciones de los archivos de entrada de pedidos (es decir, los archivos en DISTLIB) cuyos nombres empiezan por ORD. ORD* es el nombre genérico.

```
DSPOBJD   OBJ(DISTLIB/ORD*)  OBJTYPE(*FILE) +
          DETAIL(*BASIC)  OUTPUT(*)
```

La pantalla básica que resulta es la siguiente:

```
                      Visualizar Descripción de Objeto - Básica
                                                          Biblioteca 1 de 1
Biblioteca:  DISTLIB

Teclee opciones, pulse Intro

5=Visualizar atributos completos   8=Visualizar atributos de servicio


Opc  Objeto      Tipo      Atributo                 Tamaño  Texto
 _   ORDDTLP     *FILE        PF                     8192   Detalle pedido
 _   ORDHDRP     *FILE        PF                     8192   Cabecera pedido
                                                                    Final
F3=Salir  F12=Cancelar   F17=Principio   F18=Final
```

Si especifica *FULL en lugar de *BASIC o si entra un 5 delante de ORDDTLP en la pantalla básica, la pantalla completa que resulta es ésta:

```
                      Visualizar Descripción Objeto - Completa
                                                          Biblioteca 1 de 1
Objeto . . . . . :    ORDDTLP        Atributo . . . . . :   PF
```

```
  Biblioteca . . . . :      DISTLIB      Propietario  . . . . :   QSECOFR
Tipo . . . . . . . . :    *FILE          Grupo primario       :   *NONE
Información definida por el usuario:
   Atributo  . . . . . . . . . . . . :
   Texto . . . . . . . . . . . . . . :
Información de creación:
  Fecha y hora de creación . . . . . :   06/08/89  10:17:03
  Creado por el usuario  . . . . . . :   QSECOFR
  Sistema en el que fue creado . . . :   RCHAS790
  Dominio de objeto  . . . . . . . . :   *SYSTEM
Información Cambios/Utilización:
  Cambiar Fecha/hora . . . . . . . . :   05/11/90  10:03:02
  Recogida de datos de utilización . :   YES
  Fecha de última utilización  . . . :   05/11/90
  Cuenta de días de utilizados . . . :   20
  Fecha de restauración de cuenta
  de utilización . . . . . . . . . . :   03/10/90
  Permitir cambio por programa . . . :   YES
Información de auditoría:
  Valor de auditoría de objeto . . . :   *NONE
Pulse Intro para continuar.
F3=Salir  F12=Cancelar
                                        (C) COPYRIGHT IBM CORP. 1980, 1993.
```

```
                     Visualizar Descripción Objeto - Completa
                                                          Biblioteca 1 de 1
Objeto . . . . . . . :    ORDDTLP        Atributo . . . . . :   PF
  Biblioteca . . . . :      DISTLIB      Propietario  . . . . :   QSECOFR
Tipo . . . . . . . . :    *FILE

Información de almacenamiento:
  Tamaño . . . . . . . . . . . . . . :   8192
  Tamaño fuera de línea  . . . . . . :   0
  Liberado . . . . . . . . . . . . . :   NO
  Comprimido . . . . . . . . . . . . :   NO
  Agrupación almacenamiento auxiliar :   1
    Objeto desbordado  . . . . . . . :   NO
Información Salvar/Restaurar
  Salvar Fecha/hora  . . . . . . . . :
  Restaurar Fecha/hora . . . . . . . :
  Salvar mandato . . . . . . . . . . :
  Tipo de dispositivo  . . . . . . . :
                                                                    Fin
Pulse Intro para continuar.


F3=Salir  F12=Cancelar

                                        (C) COPYRIGHT IBM CORP. 1980, 1993.
```

4.8 Recuperación de Descripciones de Objeto

Puede utilizar el mandato Recuperar Descripción de Objeto (RTVOBJD) para devolver las descripciones de un objeto determinado a un procedimiento CL. Para devolver las descripciones se utilizan variables. Estas descripciones le pueden ayudar a detectar objetos en el sistema que no se utilizan.

Las descripciones que pueden devolverse como variables para un objeto son:

- Nombre de la biblioteca que contiene el objeto
- Cualquier atributo ampliado de un objeto (tal como un programa o tipo de archivo)
- Atributo definido por el usuario
- Descripción del texto del objeto
- Nombre del perfil de usuario del propietario del objeto
- Nombre del grupo primario para el objeto
- ID de agrupación de almacenamiento auxiliar
- Indicación de si el objeto ha desbordado o no el ASP en el que reside
- Fecha y hora de creación del objeto
- Fecha y hora de la última modificación del objeto
- Fecha y hora en que se salvó el objeto por última vez
- Fecha y hora en que se salvó el objeto durante una operación de salvar SAVACT (*LIB, *SYSDFN o *YES)
- Fecha y hora de la última restauración del objeto
- Nombre del perfil de usuario del creador del objeto
- Sistema en el que se ha creado el objeto
- Dominio del objeto
- Si se han recogido o no datos de utilización
- Fecha en que el objeto se utilizó por última vez
- Cuenta (número) de días que el objeto fue utilizado
- Fecha en que la cuenta de utilización se restauró por última vez
- Estado de almacenamiento de los datos del objeto
- Estado de compresión del objeto
- Tamaño del objeto en bytes
- Tamaño del objeto en bytes de almacenamiento en el momento de la última operación de salvar
- Mandato utilizado para salvar el objeto
- Número de secuencia de cinta generado cuando el objeto se salvó en cinta
- Volumen de cinta o disquete utilizado para salvar el objeto
- Tipo de dispositivo en el que el objeto se salvó por última vez
- Nombre del archivo de salvar si el objeto se salvó en un archivo de salvar
- Nombre de la biblioteca que contiene el archivo de salvar si el objeto se salvó en un archivo de salvar
- Etiqueta de archivo utilizada cuando el objeto se salvó
- Nombre del archivo fuente que se utilizó para crear el objeto
- Nombre de la biblioteca que contiene el archivo fuente que se utilizó para crear el objeto
- Nombre del miembro en el archivo fuente
- Fecha y hora en que el miembro del archivo fuente se actualizó por última vez
- Nivel del sistema operativo en que se creó el objeto
- Identificador del programa bajo licencia, nivel de release y nivel de modificación del compilador
- Nivel de control de objeto para el objeto creado
- Información sobre si la API Cambiar Descripción de Objeto (QLICOBJDD) puede modificar el objeto.
- Indicación de si el objeto se ha modificado o no con la API Cambiar Descripción de Objeto (QLICOBJD)
- Información sobre si el usuario modificó el programa o no
- Nombre, nivel de release y de modificación del programa bajo licencia si el objeto recuperado forma parte de un programa bajo licencia
- Número de Arreglo Temporal del Programa (PTF) que dio como resultado la creación del objeto recuperado
- Identificación del Informe Autorizado de Análisis de Programa (APAR)
- Tipo de auditoría para el objeto

Subtemas

4.8.1 Ejemplo de RTVOBJD

En el siguiente procedimiento CL, un mandato RTVOBJD recupera la descripción de un objeto específico. Suponga que en la biblioteca actual (MYLIB) hay un objeto llamado MOBJ.

```
  DCL  &LIB      TYPE(*CHAR) LEN(10)
  DCL  &CRTDATE  TYPE(*CHAR) LEN(13)
  DCL  &USEDATE  TYPE(*CHAR) LEN(13)
  DCL  &USECNT   TYPE(*DEC)  LEN(5 0)
  DCL  &RESET    TYPE(*CHAR) LEN(13)
  .
  .
  .
  RTVOBJD    OBJ(MYLIB/MOBJ) OBJTYPE(*FILE) RTNLIB(&LIB)
             CRTDATE(&CRTDATE) USEDATE(&USEDATE)
             USECOUNT(&USECNT) RESETDATE(&RESET)
```

Se devuelve la información siguiente al programa:

- El nombre de biblioteca actual (MYLIB) se coloca en la variable CL llamada &LIB.
- La fecha de creación de MOBJ se coloca en la variable CL denominada &CRTDATE.
- La fecha en que se utilizó por última vez MOBJ se coloca en la variable CL denominada &USEDATE.
- El número de días que MOBJ ha sido utilizado se coloca en la variable CL denominada &USECNT. La fecha de inicio de esta cuenta es el valor que se coloca en la variable CL denominada &RESET.

4.9 Creación de Información para Objetos

La siguiente información se proporciona en la descripción de objeto y se establece al crear el objeto. Es especialmente útil para la gestión y mantenimiento de objetos.

- Creador del objeto

  - El creador del objeto es el perfil de usuario que ejecuta la operación de crear. Esto es así aunque el perfil de usuario tenga un perfil de grupo y este perfil de grupo sea el propietario del objeto.

  - El creador del objeto no cambia cuando cambia la propiedad.

  - El creador es el creador del objeto en el soporte de almacenamiento cuando se restaura un objeto.

  - El creador del objeto es el usuario que ejecuta el mandato cuando se duplica un objeto utilizando el mandato Crear Objeto Duplicado (CRTDUPOBJ).

  - El creador es ***IBM** para objetos suministrados por IBM.

  - Para usuarios de objeto que ya existían en el sistema antes de la Versión 1, Release 3.0, el creador del objeto está en blanco.

- Sistema en el que se creó el objeto

  - Cuando se restaura un objeto, el sistema en el que se creó es el sistema en que el objeto ha sido creado en el soporte magnético.

  - Para los objetos suministrados por IBM, el sistema en el que se ha creado **00000000**.

  - Para los objetos que ya existían en el sistema antes de instalar la Versión 1, Release 3.0, el sistema en el que se creó está en blanco.

4.10 Detección de Objetos No Utilizados en el Sistema

La información facilitada en la descripción de los objetos puede ayudarle a detectar y gestionar los objetos que no se utilizan en el sistema.

Para detectar un objeto que no ha sido utilizado, compruebe la fecha en que se utilizó por última vez y la fecha en que se realizó la última modificación. Los mandatos Cambiar no actualizan la fecha del último uso a menos que los mismos mandatos supriman y vuelvan a crear el objeto, o que la operación de cambiar haga que el objeto se lea como parte del propio cambio.

- Fecha y hora de la última modificación

  - Cuando se crea o modifica un objeto, la hora del sistema se marca en el objeto, indicando la fecha y la hora en que tuvo lugar la modificación.

- Fecha de la última utilización

  - La fecha de la última utilización se actualiza solamente una vez al día (la primera vez que un objeto se utiliza en un día). Se utiliza la fecha del sistema.

  - Un intento fallido de utilizar un objeto no actualiza la fecha de la última utilización. Por ejemplo, si un usuario trata de utilizar un objeto al cual no está autorizado, la fecha de la última utilización no se modifica.

  - La fecha de la última utilización está en blanco para los objetos nuevos.

  - Cuando se restaura un objeto que ya existe en sistema, la fecha de la última utilización viene del objeto en el sistema. Si no existe todavía al restaurarse, la fecha permanece en blanco.

  - Los objetos que se suprimen y se vuelven a crear durante el proceso de restauración pierden la fecha de la última utilización.

  - La última fecha utilizada para un archivo de base de datos **no** se actualiza cuando el número de miembros en el archivo es cero. Por ejemplo, si el usuario utiliza CRTDUPOBJ para copiar objetos, y no hay miembros en el archivo de base de datos, la última fecha utilizada no se actualiza.

  - La última fecha de utilización para un archivo de base de datos es la última fecha de utilización del miembro del archivo con la fecha de última utilización más actual.

  - Para archivos lógicos, la fecha de la última utilización es la última vez que se utilizó un miembro lógico (o cursor).

  - Para archivos físicos, la fecha de la última utilización es la última vez que se utilizaron los datos de un espacio de datos mediante un acceso físico o lógico.

La Tabla 4-3 contiene información adicional acerca de operaciones que provocan la actualización de la fecha de último cambio para diversos tipos de objeto.

Tabla 4-3. Actualización de información de utilización

Tipo de objeto	**Mandatos y operaciones**
Todos los tipos de objeto	Mandato Otorgar Autorización sobre Objeto (CRTDUPOBJ) y otros mandatos, como Copiar Biblioteca (CPYLIB), que utilizan CRTDUPOBJ para copiar objetos. Mandato Otorgar Autorización sobre Objeto (GRTOBJAUT) (para objetos a los que se hace referencia)
Máquina AS/400 Advanced 36	Mandato CHGM36, cambiar máquina S/36* Ejecución actual de una máquina S/36 (se ejecuta cada día, se actualiza el número total de utilización)
Configuración de	CRTM36CFG (cuando se especifica FROMM36CFG),

máquina AS/400 Advanced 36	CHGM36CFG, DSPM36CFG, STRM36
Cambiar Descripción de Petición	Cambiar Actividad de Petición de Cambio de Mandato (CHGCMDCRQA)
Formato de diagrama	Mandato Visualizar Diagrama (DSPCHT)
Descripción de escenario C	Mandato Recuperar Fuente de Descripción de Escenario C mandato (RTVCLDSRC) o cuando se le hace referencia en un programa C.
Clase	Cuando se utiliza para iniciar un trabajo
Mandato (modificar desde cmd definido por el Sistema)	Al ejecutarse Al compilarse en un programa CL Cuando se solicita durante la entrada del fuente SEU Al llamar al sistema en modalidad de comprobación **Nota:** La solicitud desde la línea de mandatos y el posterior uso de F3 no se cuenta como una utilización de un mandato definido por el sistema.
Información Complementaria de Comunicaciones (CSI)	Cuando se utiliza la llamada CPI-Comunicaciones Inicializar Conversación (CMINIT) para inicializar valores para varias características de conversación desde un objeto de información complementaria.
Lista de conexiones	Cuando la lista de conexiones va más allá del estado de activación pendiente
Mapa de Cross System Product	Cuando se le hace referencia en una aplicación CSP
Tabla de Cross System Product	Cuando se le hace referencia en una aplicación CSP
Descripción de controlador	Cuando el controlador va más allá del estado de activación pendiente
Descripción de dispositivo	Cuando el dispositivo va más allá del estado de activación pendiente
Área de datos	Mandato Recuperar Área de Datos (RTVDTAARA) Mandato Visualizar Área de Datos (DSPDTAARA)
Cola de datos	La información de utilización para las API siguientes sólo se actualizan una vez por trabajo (la primera vez que se inicia una de las API). API Enviar Cola de Datos (QSNDDTAQ) API Recibir Cola de Datos (QRCVDTAQ) API Recuperar Cola de Datos (QMHQRDQD) API Leer Cola de Datos (QMHRDQM) API
Archivo (archivo de base de datos solamente a menos que se especifique lo contrario)	Al cerrarse (otros archivos, tales como un archivo de dispositivo y de salvar y también actualizados al cerrarse) Al borrarse Al inicializarse Al reorganizarse Mandatos: □ Mandato Aplicar Cambios Registrados por Diario (APYJRNCHG) □ Mandato Eliminar Cambios Registrados por Diario (RMVJRNCHG)

Recurso de font	Cuando se le hace referencia durante una operación de impresión
Definición del formulario	Cuando se le hace referencia durante una operación de impresión
Conjunto de símbolos gráficos	Cuando un programa de aplicación de gráficos PGR o GDDM* le hace referencia Al cargarlo internamente o utilizando GSLSS
Descripción del trabajo	Cuando se utiliza para establecer un trabajo
Planificación de trabajos	Cuando el sistema somete un trabajo para una entrada de planificación de trabajos
Cola de trabajo	Cuando una entrada se coloca o se elimina de la cola
Descripción de línea	Cuando la línea va más allá del estado de activación pendiente
Escenario	API de recuperación de escenario QLGRTVLC Cuando un trabajo arranca, si el valor LOCALE de perfil de usuario contiene un nombre de vía para un objeto *LOCALE válido.
Menú	Cuando se visualiza un menú utilizando el mandato GO
Archivos de mensajes	Cuando se recupera un mensaje de un archivo de mensajes que no es QCPFMSG, ##MSG1, ##MSG2 ni QSSPMSG (tal como cuando se crean unas anotaciones de trabajo, se visualiza una cola de mensajes, se solicita ayuda en un mensaje en la anotación QHST, o un programa recibe un mensaje diferente de un mensaje con marca) Mandato Fusionar Archivo de Mensajes (MRGMSGF) excepto cuando el archivo de mensajes es QCPFMSG, ##MSG1, ##MSG2 o QSSPMSG
Cola de mensajes	Cuando un mensaje se envía, se recibe o se lista en una cola de mensajes que no es QSYSOPR ni QHST
Descripción de interfaz de red	Cuando la descripción de interfaz de red va más allá del estado de activación pendiente
Lista de Nodos	Actualizada solamente por los mandatos y operaciones que afectan a todos los tipos de objeto
Cola de salida	Cuando una entrada se coloca o se elimina de la cola
Recubrimiento	Cuando se le hace referencia durante una operación de impresión
Definición de página	Cuando se le hace referencia durante una operación de impresión
Segmento de página	Cuando se le hace referencia durante una operación de impresión
Grupo de paneles	Cuando se utiliza la tecla Ayuda para solicitar información de ayuda para una solicitud o un panel determinado, la fecha de utilización se actualiza Cuando un panel se visualiza o se imprime desde un grupo de paneles
Grupo descriptor de impresión	Cuando se le hace referencia durante una operación de impresión
Disponibilidad del Producto	Actualizada solamente por los mandatos y operaciones que afectan a todos los tipos de objeto
Carga de Producto	Actualizada solamente por los mandatos y operaciones que afectan a todos los tipos de objeto

Programa	Mandato Recuperar Fuente CL (RTVCLSRC) Cuando se ejecuta y no es un programa del sistema Cuando se comunica con otro programa para crear un objeto de programa
Configuración PSF	Cuando se le hace referencia durante una operación de impresión
Definición de consulta	Cuando se utiliza para generar un informe Cuando se extrae o exporta
Formulario de gestor de consultas	Cuando se utiliza para generar un informe Cuando se extrae o exporta
Consulta de gestor de consultas	Cuando se utiliza para generar un informe Cuando se extrae o exporta
Índice de búsqueda	Cuando se utiliza la tecla F11 en la información de ayuda en línea Cuando se utiliza el mandato Iniciar Índice de Búsqueda (STRSCHIDX)
Almacenamiento de Servidor	Activar/Desactivar Configuración (VRYCFG) se ejecuta sobre un objeto de descripción de servidor de red
Paquete SQL	Actualizada solamente por los mandatos y operaciones que afectan a todos los tipos de objeto
Descripción del subsistema	Al arrancarse un subsistema
Diccionario de ayuda ortográfica	Cuando se utiliza para crear otro diccionario Al recuperar Cuando se encuentra una palabra en el diccionario durante una comprobación ortográfica y el diccionario no es un diccionario de ayuda ortográfica suministrado por IBM
Tabla	Utilizada por un programa para conversión
Perfil de usuario	Cuando se inicia un trabajo para el perfil Cuando el perfil es un perfil de grupo y el trabajo se arranca utilizando un miembro del grupo Mandato Otorgar Autorización a Usuario (GRTUSRAUT) (para un perfil al que se hace referencia)
Personalización del Usuario de la Estación de Trabajo	Actualizada solamente por los mandatos y operaciones que afectan a todos los tipos de objeto Cuando el usuario elige la 'Opción 10 = Someter inmediatamente' en el panel WRKJOBSCDE

A continuación se presenta información adicional de utilización de objetos proporcionada en la descripción del objeto:

- Contador del número de días que se ha utilizado el objeto

  - La cuenta aumenta cuando se actualiza la fecha de la última utilización.

  - Cuando un objeto que ya existe en el sistema se restaura, el número de días que se ha utilizado proviene del objeto en el sistema. Si no existe cuando se restaura, la cuenta es cero.

  - Los objetos que se suprimen y se vuelven a crear durante el proceso de restauración pierden el cómputo de días que han sido utilizados.

  - Para los objetos nuevos, la cuenta de días que se han utilizado es

cero.

Nota: El sistema AS/400 *no puede* determinar la diferencia entre archivos de dispositivos antiguos y nuevos. Si restaura un archivo de dispositivos en el sistema y ya existe uno con el mismo nombre, suprima éste último en caso de que desee restablecer a cero el cómputo de días de utilización. Si el archivo no se suprime, el sistema interpretará esta acción como una operación de restauración de un objeto antiguo y mantendrá el cómputo de días de utilización.

- La cuenta de días de utilización para un archivo de base de datos es la suma de las cuentas de días de utilización para todos los miembros del archivo. Si hay un desbordamiento en la suma, se muestra el valor máximo (del campo cuenta de días utilizados).

- Fecha en que se restauró la cuenta de días que se utilizó el objeto

  - Cuando la cuenta de días que se ha utilizado el objeto se restaura utilizando el mandato Cambiar Descripción de Objeto (CHGOBJD) o la API Cambiar Descripción de Objeto (QLICOBJD), la fecha se registra. De este modo, el usuario sabe cuánto tiempo ha estado activa la cuenta de días que se ha utilizado el objeto.

  - Cuando se restaura la cuenta de días de utilización para un archivo, se restaura la cuenta de días de utilización de todos los miembros.

Las situaciones habituales que pueden suprimir el computo de días de utilización y la fecha de última utilización son las siguientes:

- Se dañan objetos del sistema y se restauran.

- Se restauran programas cuando el sistema no se encuentra en un estado restringido.

Puede utilizar el mandato Visualizar Descripción de Objeto (DSPOBJD) para visualizar una descripción completa de un objeto. Puede utilizar el mismo mandato para grabar la descripción en un archivo de salida. Para recuperar las descripciones, utilice el mandato Recuperar Descripción de Objeto (RTVOBJD).

Nota: Existe un interfaz de programación de aplicaciones (API), QSUROBJD, que proporciona la misma información que el mandato Recuperar Descripción de Objeto. Para mas información, consulte el manual System API Reference.

El mandato Recuperar Descripción del Miembro (RTVMBRD) y el mandato Visualizar Descripción de Archivo (DSPFD) facilitan una información similar para los miembros de un archivo.

La información sobre la utilización de objetos no se actualiza en los siguientes tipos de objeto:

- Tabla de alerta
- Lista de autorizaciones
- Lista de configuración
- Descripiones de clase de servicio
- Diccionarios de datos
- Documento
- Diccionario de juego de caracteres de doble byte
- Clasificación de juego de caracteres de doble byte
- Tabla de juego de caracteres de doble byte
- Descripciones de edición
- Registro de salida
- Filtro
- Tabla de control de formularios
- Carpeta
- Descripción de Paquete de Intercambio Internet
- Diario
- Receptor de diarios
- Biblioteca
- Descripción de modalidad
- Descripción de Servidor de Red
- Descripción NetBIOS
- Tabla de referencia de objeto
- Definición de producto
- Tablas de conversión de códigos de referencia
- Autentificación de servidor
- Descripción de sesión

- Descripción de máquina S/36
- Cola de usuario

4.11 Traslado de Objetos de una Biblioteca a Otra

Puede utilizar el mandato Mover Objeto (MOVOBJ) para trasladar objetos entre bibliotecas. El traslado de objetos de una biblioteca a otra resulta útil porque puede hacer que un objeto no esté disponible temporalmente y le permite sustituir una versión antigua de un objeto por una nueva. Por ejemplo, se puede crear un archivo primario nuevo para colocarlo temporalmente en una biblioteca que no es la que contiene el archivo primario antiguo. Puesto que los datos del archivo antiguo normalmente se copian en el archivo primario nuevo, el archivo antiguo no se puede suprimir hasta haber creado el nuevo. Después el archivo primario antiguo podrá suprimirse y el nuevo trasladarse a la biblioteca que contenía el archivo antiguo.

Solamente puede mover un objeto si dispone de autorización de gestión de objetos para este objeto, autorización de supresión o de ejecución para la biblioteca desde la que se desplaza el objeto, y autorización de adición y lectura sobre la biblioteca a la que se traslada el objeto.

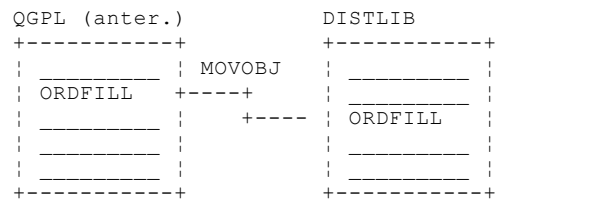
Puede sacar un objeto de una biblioteca temporal, QTEMP, pero no puede trasladar un objeto a QTEMP. Asimismo, no puede mover una cola de salida a menos que esté vacía.

El traslado de diarios y receptores de diario se limita a trasladar dichos tipos de objetos a la biblioteca en la que se crearon originariamente. Si los objetos de diario se han colocado en QRCL mediante un mandato Reclamar Almacenamiento (RCLSTG), debe trasladarlos de nuevo a su biblioteca original para que sean operativos.

A continuación figura una lista de objetos que *no se pueden* mover:

- Listas de autorización
- Descripciones de clase de servicio
- Listas de configuración
- Listas de conexiones
- Descripciones de controlador
- Diccionarios de datos
- Tablas de font de Juego de Caracteres de Doble Byte (DBCS)
- Descripciones de dispositivo
- Colas de mensajes de estación de pantalla
- Documentos
- Listas de documentos
- Descripciones de edición
- Registro de salida
- Carpetas
- Descripción de Paquete de Intercambio Internet
- Planificaciones de trabajo
- Bibliotecas
- Descripciones de línea
- Descripciones de modalidad
- Descripción NetBIOS
- Descripciones de interfaz de red
- Paquetes SQL
- Descripciones de máquina del Sistema/36
- Anotaciones históricas del sistema (QHST)
- Cola de mensajes del operador del sistema (QSYSOPR)
- Perfiles de usuario

En el ejemplo siguiente, se traslada un archivo desde QGPL (donde se colocó cuando se creó) a la biblioteca de entrada de pedidos DISTLIB de forma que se agrupe con otros archivos de entrada de pedidos.



Para mover el objeto, debe especificar la biblioteca de destino (TOLIB) y el tipo de objeto (OBJTYPE):

```
MOVOBJ  OBJ(QGPL/ORDFILL)  OBJTYPE(*FILE)  TOLIB(DISTLIB)
```

Cuando se mueven objetos, hay que tener cuidado de no trasladar aquellos objetos de los cuales dependen otros. Por ejemplo, puede que los procedimientos CL dependan de las definiciones de mandato de los mandatos utilizados en el procedimiento para que estén en la misma biblioteca en el momento de la ejecución tal como estaban en tiempo de creación de módulo.

Las definiciones de mandato, en tiempo de compilación y de ejecución, se encuentran en la biblioteca especificada o en una biblioteca de la lista de bibliotecas si se especificó *LIBL. Si se especifica un nombre de biblioteca, al ejecutar el programa las definiciones de mandatos deben estar en la misma biblioteca en que estaban al crearlo. Si se especifica *LIBL, las definiciones de mandatos pueden moverse desde la creación del programa hasta su ejecución, siempre y cuando se trasladen a una biblioteca de la lista de bibliotecas. Asimismo, cualquier programa de aplicación que escriba puede depender de que ciertos objetos estén en bibliotecas determinadas.

Un objeto que hace referencia a otro puede depender de la ubicación de éste último (aunque se puede especificar *LIBL como ubicación del mismo). Por lo tanto, si mueve un objeto, debe cambiar las referencias a él en otros objetos. La lista siguiente muestra ejemplos de objetos que hacen referencia a otros objetos:

- Las descripciones del subsistema hacen referencia a colas de trabajo, clases, colas de mensajes y programas.
- Las definiciones de mandato hacen referencia a programas y archivos de mensajes.
- Los archivos de dispositivo hacen referencia a colas de salida.
- Las descripciones de dispositivo hacen referencia a tablas de conversión.
- Las descripciones de trabajo hacen referencia a colas de trabajo y colas de salida.
- Los archivos de base de datos hacen referencia a otros archivos de base de datos.
- Los archivos lógicos hacen referencia a archivos físicos o selecciones de formato.
- Los perfiles de usuario hacen referencia a programas, menús, descripciones de trabajo, colas de mensajes y colas de salida.
- Los programas CL hacen referencia a archivos de pantalla, áreas de datos y otros programas.
- Los archivos de pantalla hacen referencia a archivos de base de datos.
- Los archivos de impresora hacen referencia a las colas de salida.

Nota: Debe tener cuidado cuando mueva objetos desde la biblioteca del sistema QSYS. Estos objetos son necesarios para que el sistema funcione adecuadamente y el sistema debe poder encontrar los objetos. Esto también es cierto para algunos de los objetos de la biblioteca general QGPL, en especial para colas de salida y trabajos.

El mandato MOVOBJ traslada un sólo objeto a la vez. Para ver un ejemplo de traslado de varios objetos en un solo mandato, vea el mandato MOVLIBOBJ en QUSRTOOL.

4.12 Creación de Objetos Duplicados

Puede utilizar el mandato Crear Objeto Duplicado (CRTDUPOBJ) para crear una copia de un objeto que ya existe. El objeto duplicado tiene el mismo tipo y autorización y se crea en la misma agrupación de almacenamiento auxiliar (ASP) que el objeto original. El usuario que emite el mandato es el propietario del objeto duplicado.

Notas:

1. Si crea un objeto duplicado de un archivo registrado por diario, el objeto duplicado (archivo) no tendrá el registro por diario activo. Sin embargo, posteriormente puede seleccionar el registro por diario de este objeto.
2. Si crea un objeto duplicado, y el objeto (archivo) no tiene miembros, el campo de última fecha utilizada son blancos, y la cuenta para el número de días es cero.

Puede duplicar un objeto si dispone de autorización de gestión y uso sobre el objeto, autorización de uso y adición sobre la biblioteca en la que se va a colocar el objeto duplicado, autorización de uso sobre la biblioteca en la que existe el objeto original, y autorización de adición sobre el perfil de usuario del proceso.

Para duplicar una lista de autorizaciones, hay que tener autorización de gestión de lista de autorizaciones para el objeto y autorizaciones de adición y operativa de objeto para la biblioteca QSYS.

Sólo se duplican las definiciones de colas de trabajos, colas de mensajes, colas de salida y colas de datos. Las colas de trabajos y de salida no se pueden duplicar en la biblioteca temporal (QTEMP). Para un archivo físico o de un archivo de salvar, puede especificar si los datos del archivo también se tienen que duplicar.

Los siguientes objetos *no se pueden* duplicar:

- Máquina AS/400 Advanced 36*
- Descripciones de clase de servicio
- Listas de configuración
- Listas de conexiones
- Descripciones de controlador
- Diccionarios de datos
- Descripciones de dispositivo
- Colas de datos
- Documentos
- Listas de documentos
- Descripciones de edición
- Registro de salida
- Carpetas
- Tablas de fonts DBCS
- Descripción de Paquete de Intercambio Internet
- Planificaciones de trabajo
- Diarios
- Receptores de diario
- Bibliotecas
- Descripciones de línea
- Descripciones de modalidad
- Descripciones NetBIOS
- Descripciones de interfaz de red
- Descripción de Servidor de Red
- Tablas de conversión de códigos de referencia
- Diccionarios de ayuda ortográfica
- Paquetes SQL
- Descripciones de máquina del Sistema/36
- Cola de mensajes del operador del sistema (QSYSOPR)
- Anotaciones históricas del sistema (QHST)
- Perfiles de usuario
- Colas de usuario.

En algunos casos, es posible que desee duplicar solamente algunos de los datos de un archivo haciendo que al mandato CRTDUPOBJ le siga un mandato CPYF que especifique los valores de selección.

El mandato siguiente crea una copia duplicada del archivo físico de cabecera de pedido y duplica los datos del archivo:

```
  CRTDUPOBJ OBJ(ORDHDRP) FROMLIB(DSTPRODLIB) OBJTYPE(*FILE) +
            TOLIB(DISTLIB2) NEWOBJ(*SAME) DATA(*YES)
```

Cuando cree un objeto duplicado, debe tener en cuenta las consecuencias de la creación de un duplicado de un objeto que hace referencia a otro objeto. Muchos objetos hacen referencia a otros objetos por el nombre, y una gran parte de estas referencias están calificadas por un nombre de biblioteca específico. Por lo tanto, el objeto duplicado podría contener

una referencia a un objeto que se encuentra en una biblioteca distinta de la biblioteca en la que reside el objeto duplicado. Para todos los tipos de objeto que no son archivos, las referencias a otros objetos se suplican en el objeto duplicado. En el caso de los archivos, los objetos duplicados comparten el formato del archivo original.

Cualquier archivo físico que se encuentre en la biblioteca de origen y en el que se base un archivo lógico también debe existir en la biblioteca de destino. Se comparan el nombre de formato del registro y el ID de nivel de registro de los archivos físicos de las bibliotecas de origen y de destino; si los archivos físicos no coinciden, el archivo lógico no se duplica.

Si un archivo lógico utiliza una selección de formato que existe en la biblioteca de origen, se supone que dicha selección también existe en la biblioteca de destino.

4.13 Redenominación de Objetos

Puede utilizar el mandato Redenominar Objeto (RNMOBJ) para redenominar objetos. Sin embargo, sólo puede hacerlo si dispone de autorización de gestión de objetos para el objeto y autorización de actualización y ejecución sobre la biblioteca que contiene el objeto.

Para redenominar una lista de autorizaciones, debe tener autorización de gestión de lista de autorizaciones, así como autorización de actualización y de lectura para la biblioteca QSYS.

Los siguientes objetos *no se pueden* redenominar:

- Descripciones de clase de servicio
- Diccionarios de datos
- Tablas de fonts DBCS
- Colas de mensajes de estación de pantalla
- Documentos
- Listas de documentos
- Registro de salida
- Carpetas
- Planificaciones de trabajo
- Diarios
- Receptores de diario
- Descripciones de modalidad
- Descripción de Servidor de Red
- Paquetes SQL
- Descripciones de máquina del Sistema/36
- Anotaciones históricas del sistema (QHST)
- La biblioteca del sistema, QSYS y la biblioteca temporal, QTEMP
- Cola de mensajes del operador del sistema (QSYSOPR)
- Perfiles de usuario

Asimismo, no puede redenominar una cola de salida a menos que esté vacía. No debe redenominar los mandatos suministrados por IBM porque los programas bajo licencia también los utilizan.

Para redenominar un objeto, hay que especificar el nombre actual del objeto, el nombre con el que va a redenominarse el objeto y el tipo de objeto.

El mandato siguiente RNMOBJ cambia el nombre del objeto ORDERL por ORDFILL:

```
  RNMOBJ   OBJ(QGPL/ORDERL)  OBJTYPE(*FILE)   NEWOBJ(ORDFILL)
```

No puede especificar un nombre calificado para el nuevo nombre de objeto porque el objeto permanece en la misma biblioteca. Si el objeto redenominado se está utilizando al emitir el mandato RNMOBJ, el mandato no se ejecuta y el sistema envía un mensaje.

Al redenominar los objetos, hay que tener cuidado de no redenominar objetos de los que dependen otros. Por ejemplo, los programas CL dependen de que las definiciones de los mandatos utilizados en el programa tengan el mismo nombre al ejecutar el programa que al compilarlo. Así pues, si la definición del mandato se redenomina en el tiempo que transcurre entre la creación y la ejecución del programa, éste no puede ejecutarse porque no se encontrarán los mandatos. Asimismo, cualquier programa de aplicación que se escriba depende de que ciertos objetos tengan el mismo nombre en ambas ocasiones.

No se puede redenominar una biblioteca que contenga un diario, un receptor de diario, diccionario de datos o paquete SQL.

Un objeto que hace referencia a otro puede depender de los nombres del objeto y de la biblioteca (aunque pueda especificarse *LIBL para el nombre de biblioteca). Así pues, si redenomina un objeto, hay que modificar las referencias que se hagan a él en otros objetos. Para ver una lista de los objetos que hacen referencia a otros, consulte el apartado "Traslado de Objetos de una Biblioteca a Otra" en el tema 4.11.

Si redenomina un archivo físico o lógico, no se redenominan los miembros del archivo. Sin embargo, puede utilizar el mandato Redenominar Miembro (RNMM) para redenominar un miembro de un archivo físico o lógico.

Nota: Hay que tener cuidado al redenominar los objetos de la biblioteca del sistema QSYS. Estos objetos son necesarios para que el sistema funcione adecuadamente y el sistema debe poder encontrar los objetos. Esto es también válido para algunos de los objetos de la biblioteca de uso general QGPL.

4.14 Compresión o Descompresión de Objetos

Puede utilizar el mandato Comprimir Objeto (CPROBJ) para comprimir objetos seleccionados con el fin de ahorrar espacio en disco en el sistema, y el mandato Descomprimir Objeto (DCPOBJ) para descomprimir objetos que se habían comprimido. Los tipos de objeto que se pueden comprimir o descomprimir son *PGM, *SRVPGM, *MODULE, *PNLGRP, *MENU (sólo menús UIM), y *FILE (sólo archivos de pantalla o archivos de impresión). Los archivos de base de datos no se pueden comprimir. Los objetos del cliente, así como los suministrados por el sistema AS/400, pueden comprimirse o descomprimirse. Para ver o recuperar el estado de compresión de un objeto, utilice el mandato Visualizar Descripción de Objeto (DSPOBJD) (pantalla *FULL) o Recuperar Descripción de Objeto (RTVOBJD).

Subtemas
4.14.1 Compresión de Objetos
4.14.2 Objetos Descomprimidos Temporalmente
4.14.3 Descompresión Automática de Objetos

4.14.1 Compresión de Objetos

Los tipos de objeto *PGM, *SRVPGM, *MODULE, *PNLGRP, *MENU y *FILE (sólo archivos de pantalla y de impresión) pueden comprimirse utilizando los mandatos CPROBJ o DCPOBJ. Los objetos sólo pueden comprimirse cuando se dan estas dos situaciones:

- Si el sistema puede obtener un bloqueo exclusivo sobre el objeto.
- Si el volumen comprimido ahorra espacio en disco.

Las siguientes restricciones se aplican a la compresión de objetos:

- Los programas creados antes de la Versión 1 Release 3 del sistema operativo no se pueden comprimir.
- Los programas, programas de servicio o módulos, creados antes de la Versión 3 Release 6 del sistema operativo que no se han traducido de nuevo, no se pueden comprimir.
- Los programas de las bibliotecas suministradas por IBM QSYS y QSSP no pueden comprimirse a menos que el valor de agrupación de páginas sea *BASE. Utilice el mandato Visualizar Programa (DSPPGM) para ver el valor de agrupación de páginas de un programa. Los programas de las bibliotecas que no sean QSYS y QSSP se pueden comprimir prescindiendo de su valor de agrupación de páginas.
- Sólo se pueden comprimir los menús con el atributo UIM.
- Sólo se pueden comprimir los archivos con los atributos DSPF y PRTF.
- El sistema debe estar en estado restringido (todos los subsistemas finalizados) para comprimir objetos de programa en las bibliotecas del sistema.
- Cuando se comprime un programa, éste no debe estar ejecutándose; de lo contrario, el programa finalizará de forma anormal.

La compresión es mucho más rápida si utiliza varios trabajos en estado no restringido, tal como se muestra en la tabla siguiente:

Tabla 4-4. Comprimir objetos utilizando varios trabajos

Tipo de objeto	**Suministrado por IBM**	**Suministrado por el usuario**
*FILE	Trabajo 3: QSYS	Trabajo 7: USRLIB1
*MENU	Trabajo 2: QSYS	Trabajo 8: USRLIB1
*MODULE	No se aplica	Trabajo 10: USRLIB1
*PGM	Sólo estado restringido	Trabajo 5: USRLIB1
*PNLGRP	Trabajo 1: QSYS Trabajo 4: QHLPSYS	Trabajo 6: USRLIB1
*SRVPGM	Trabajo 11: QSYS	Trabajo 9: USRLIB1

4.14.2 Objetos Descomprimidos Temporalmente

El sistema descomprime temporalmente y de forma automática los objetos comprimidos cuando los utiliza. Un objeto descomprimido temporalmente permanecerá descomprimido hasta que:

- Se efectúa una IPL del sistema. Esto provoca que un objeto descomprimido temporalmente se borre (el objeto comprimido permanece).
- Se utiliza un mandato Reclamar Almacenamiento Temporal (RCLTMPSTG) para reclamar objetos descomprimidos temporalmente. Esto hace que los objetos descomprimidos temporalmente se borren (los objetos comprimidos permanecen) si los objetos no se han utilizado durante un número determinado de días.
- El objeto descomprimido temporalmente se utiliza más de 2 días o más de 5 veces en la misma IPL, en cuyo caso quedará descomprimido permanentemente.
- Se utiliza un mandato DCPOBJ para descomprimir un objeto, en cuyo caso quedará descomprimido permanentemente.
- El sistema tenga un bloqueo exclusivo del objeto.

Notas:

1. Los objetos de tipo *PGM, *SRVPGM o *MODULE no se pueden descomprimir temporalmente. Si llama o depura un programa comprimido, éste queda descomprimido permanentemente de forma automática.

2. Los objetos de archivo comprimidos, cuando se abren, quedan descomprimidos automáticamente.

3. Si se recupera la descripción de un archivo comprimido, el archivo queda temporalmente descomprimido. Dos ejemplos de recuperación de archivo son:

   - Usar el mandato Visualizar Descripción de Campo de Archivo (DSPFFD) para visualizar la información de nivel de campo de un archivo.

   - Usar el mandato Declarar Archivo (DCLF) para declarar un archivo.

4.14.3 Descompresión Automática de Objetos

El sistema descomprime los objetos comprimidos suministrados con el OS/400 o con otros programas bajo licencia IBM *después de instalar los programas bajo licencia*. La descompresión se produce sólo si se dispone de suficiente almacenamiento en el sistema.

El sistema arranca automáticamente los trabajos del sistema nombrados QDCPOBJx para descomprimir objetos.

|El número de trabajos QDCPOBJ se basa en el número de procesadores + 1.
|Son trabajos del sistema ejecutándose con prioridad 60, que el usuario no
|puede cambiar, finalizar ni retener. Un trabajo DCPOBJx puede estar en
|uno de los siguientes estados, que pueden obtenerse con el mandato
|Trabajar con Trabajo Activo (WRKACTJOB):

|□ RUN (running): El trabajo está activo descomprimendo objetos.
|□ EVTW (event wait): El trabajo no está activo descomprimiendo objetos.
| El trabajo se activa en el caso de que necesiten descomprimirse más
| objetos (p.e. se instalan programas con licencia adicionales).
|□ DLYW (delay wait): El trabajo está temporalmente parado. Las
| siguientes situaciones podrían causar una parada de los trabajos
| QDCPOBJx :
| - El sistema está ejecutándose en estado restringido (p.e. se
| ejecutó ENDSYS o ENDSBS *ALL)
| - Se instaló recientemente un programa bajo licencia desde la
| pantalla "Trabajar con Programas bajo Licencia". El trabajo está
| en una espera retardada por un máximo de 15 minutos hasta que
| empiece a descomprimir objetos.
|□ LCKW (lock wait): El objeto está esperando un bloqueo interno.
| Típicamente, ocurre cuando un trabajo QDCPOBJ está en estado DLYW.

|Los siguientes requisitos de almacenamiento se aplican si el sistema
|operativo se instaló sobre un sistema operativo existente:

|□ El sistema debe tener más de 250 megabytes de almacenamiento no
| utilizado para que los trabajos QDCPOBJx puedan arrancar.
|□ En un sistema con un almacenamiento disponible de más de 750MB, se
someten trabajos para descomprimir todos los objetos del sistema
recién instalados.
|□ En un sistema con menos de 250 MB de almacenamiento disponible, no se
someten trabajos, y los objetos se descomprimen a medida que se
utilizan.
|□ En un sistema cuyo almacenamiento disponible sea sólo de 250MB y
| 750MB, sólo se descomprimen automáticamene los objetos que se utilizan
con frecuencia.

Los ***objetos utilizados con frecuencia*** son aquéllos que se han utilizado cinco veces como mínimo y cuya última utilización se ha producido en los últimos 14 días. Los objetos poco utilizados permanecen comprimidos.

|El sistema debe tener más de 1000MB de almacenamiento no utilizado, si el
sistema operativo se instala sobre un sistema que se ha inicializado
|utilizando 2 opciones, Instalar Código Interno bajo Licencia, y
|Inicializar el Sistema, desde la pantalla Instalar Código Interno bajo
|Licencia (LIC).

|Si los trabajos QDCPOBJx están activos en la última finalización del
|sistema, los trabajos se arrancan de nuevo en el siguiente IPL.

4.15 Supresión de Objetos

Para suprimir un objeto, puede utilizar el mandato de supresión (DLTxxx) que corresponda al tipo de objeto o bien puede utilizar la opción de suprimir en la pantalla Trabajar con Objetos (que aparece desde la pantalla Trabajar con Bibliotecas (WRKLIB)). Para suprimir un objeto, debe tener autorización de existencia de objetos sobre el objeto y autorización de ejecución sobre la biblioteca. Solamente el propietario de una lista de autorizaciones o un usuario con la autorización especial *ALLOBJ puede suprimir la lista de autorizaciones.

Cuando se suprime un objeto, debe estar seguro de que nadie más necesita ni utiliza el objeto. Generalmente, si otro usuario lo está utilizando, el objeto no se puede suprimir. Sin embargo, los programas se pueden suprimir a menos que utilice el mandato Asignar Objeto (ALCOBJ) para asignar el programa antes de que se le llame.

Los mandatos para crear programas y archivos de dispositivos disponen del parámetro REPLACE, que permite a los usuarios seguir utilizando la versión antigua de un programa que se ha sustituido.

Las versiones antiguas de estos programas que se han vuelto a crear están almacenadas en la biblioteca QRPLOBJ.

Debe tener cuidado de no borrar los objetos que se encuentran en las bibliotecas del sistema, ya que son necesarios para el correcto funcionamiento del sistema.

En la mayoría de mandatos de supresión, puede especificar un nombre genérico en lugar de un nombre de objeto. Antes de utilizar una supresión genérica, es posible que desee especificar el nombre genérico, utilizando el mandato DSPOBJD, para verificar que la supresión genérica *sólo* suprimirá los objetos que desea eliminar. Véase el apartado "Utilización de Nombre de Objeto Genéricos" en el tema 4.3.3 para obtener más información sobre la especificación genérica de objetos.

Para obtener información sobre la supresión de bibliotecas, véase el apartado "Supresión y Borrado de Bibliotecas" en el tema 4.4.7.

4.16 Asignación de Recursos

Los objetos se asignan en el sistema para garantizar la integridad y conseguir el más alto grado de concurrencia. Un objeto estará protegido aunque se realicen varias operaciones en él al mismo tiempo. Por ejemplo, un objeto se asigna de manera que sólo puedan leerlo dos usuarios simultáneamente o que sólo un usuario pueda leerlo mientras que otro puede leerlo y actualizarlo.

El OS/400 asigna objetos mediante la función que se está realizando en el objeto. Por ejemplo:

- Si un usuario visualiza o vuelca un objeto, otro usuario puede leerlo.
- Si un usuario está modificando, suprimiendo, cambiando el nombre o moviendo un objeto, nadie más puede utilizarlo.
- Si un usuario está salvando un objeto, otro usuario puede leerlo, pero no actualizarlo ni suprimirlo; si un usuario está restaurando el objeto, nadie más puede leerlo ni actualizarlo.
- Si un usuario está abriendo un archivo de base de datos para entrada, otro usuario puede leerlo. Si un usuario está abriendo un archivo de base de datos para la salida, otro usuario puede actualizarlo.
- Si un usuario está abriendo un archivo de dispositivo, otro usuario sólo podrá leerlo.

Normalmente, los objetos se asignan bajo petición; es decir, cuando un paso de trabajo necesita un objeto, lo asigna, lo utiliza y lo desasigna para que otro trabajo pueda utilizarlo. El primer trabajo que solicita el objeto se lo asigna. En su programa, puede manejar las excepciones que se producen si un objeto no puede asignarse de acuerdo con la petición. Véase el Capítulo 7 y el Capítulo 8 para obtener más información sobre la supervisión de mensajes o el manual de consulta de lenguajes de alto nivel para obtener información sobre el manejo de excepciones.

En algunas ocasiones deseará asignar un objeto para un trabajo antes de que éste precise dicho objeto para asegurar su disponibilidad, de manera que una función que se hubiera completado parcialmente no tendría que esperar a un objeto. A esto se le denomina preasignar un objeto. Puede preasignar objetos mediante el mandato Asignar Objeto (ALCOBJ).

Los objetos se asignan dependiendo tanto de la función para la cual fueron creados (lectura y actualización) como de si pueden ser compartidos (utilizados por más de un trabajo). Al archivo y al miembro siempre se les asigna *SHRRD y los datos del archivo se asignan con la etiqueta de bloqueo especificada con el estado de bloqueo. Un estado de bloqueo identifica tanto la utilización del objeto como si éste está compartido. Los cinco estados de bloqueo son los siguiente (los valores de los parámetros aparecen entre paréntesis):

- Exclusivo (*EXCL). El objeto está reservado para el uso exclusivo por parte del trabajo que lo solicita; ningún otro trabajo puede utilizar el objeto. Sin embargo, si el objeto ya ha sido asignado a otro trabajo, el suyo ya no puede tener un uso exclusivo de dicho objeto. Este estado de bloqueo es apropiado cuando un usuario no desea que ningún otro tenga acceso al objeto hasta que finalice la función que se está realizando en esos momentos.

- Exclusivo con permiso de lectura (*EXCLRD). El objeto está asignado al trabajo que lo ha solicitado, pero otros trabajos pueden leerlo. Este bloqueo es adecuado cuando un usuario desea impedir que otros usuarios efectúen una operación que no sea de lectura.

- Compartido para actualización (*SHRUPD). El objeto puede compartirse con otro trabajo para la actualización y para la lectura. Es decir, otro usuario puede solicitar para el mismo objeto un estado de bloqueo compartido para actualización o compartido para lectura. Este estado de bloqueo es apropiado cuando un usuario trata de modificar un objeto pero permite que otros usuarios tengan capacidad para leer o modificar el mismo objeto.

- Compartido sin actualización (*SHRNUP). El objeto se puede compartir con otro trabajo si éste solicita un estado de bloqueo compartido sin actualización o compartido con lectura. Este estado de bloqueo es apropiado cuando un usuario no va a modificar un objeto pero desea asegurarse de que ningún otro usuario podrá hacerlo.

- Compartido para lectura (*SHRRD). El objeto se puede compartir con otro trabajo si el usuario no solicita un uso exclusivo del objeto. Es decir, otro usuario puede solicitar un estado de bloqueo exclusivo con permiso de lectura, compartido para actualización, compartido para lectura o compartido sin actualización.

Nota: La asignación de una biblioteca no limita las operaciones que pueden realizarse en los objetos que hay dentro de la biblioteca.

Es decir, si un trabajo coloca en una biblioteca un estado de bloqueo exclusivo con permiso de lectura o compartido con actualización, otros trabajos ya no podrán colocar objetos en la biblioteca ni eliminarlos de ella; sin embargo, los otros trabajos sí pueden actualizar objetos de la biblioteca.

La siguiente tabla muestra las combinaciones válidas de estado de bloqueo para un objeto:

Tabla 4-5. Combinaciones Válidas de Estado de Bloqueo

Si un trabajo obtiene este estado de bloqueo:	**Otro trabajo puede obtener este estado de bloqueo:**
*EXCL	Ninguno
*EXCLRD	*SHRRD
*SHRUPD	*SHRUPD o *SHRRD
*SHRNUP	*SHRNUP o *SHRRD
*SHRRD	*EXCLRD, *SHRUPD, *SHRNUP o *SHRRD

Puede especificar todos estos tipos de estado de bloqueo (*EXCL, *EXCLRD, SHRUPD, SHRNUP y SHRRD) para la mayoría de tipos de objeto. Esto no se aplica a *todos* los tipos de objeto. Los tipos de objeto que **no** pueden tener especificados los cinco estados de bloqueo se listan en la siguiente tabla con los estados de bloqueo válidos para el tipo de objeto:

Tabla 4-6. Estados de Bloqueo Válido para Tipos de Objeto Específicos

Tipo de objeto	***EXCL**	***EXCLRD**	***SHRUPD**	***SHRNUP**	***SHRRD**
Descripción de dispositivo		x			
Biblioteca		x	x	x	x
Cola de mensajes	x				x
Grupo de paneles	x	x			
Programa	x	x			x
Descripción del subsistema	x				

Para asignar un objeto, debe tener autorización de existencia de objetos, de gestión de objetos o autorización operativa para el objeto. Los objetos asignados se desasignan automáticamente al finalizar un paso de direccionamiento. Para desasignar un objeto en cualquier otro momento, utilice el mandato Desasignar Objeto (DLCOBJ).

Puede asignar un programa antes de llamarlo para impedir su supresión. Para evitar que un programa se ejecute en distintos trabajos al mismo tiempo, hay que colocar un bloqueo exclusivo en el programa en cada trabajo antes de llamar al programa en cualquier trabajo. Los siguientes objetos no se pueden asignar con el mandato ALCOBJ:

- Máquina AS/400 Advanced 36
- Configuración de máquina AS/400 Advanced 36

No puede utilizar los mandatos ALCOBJ o DLCOBJ para asignar una descripción de dispositivo APPC.

A continuación se facilita un ejemplo de trabajo de proceso por lotes que necesita dos miembros de archivo para la actualización. Los miembros de ambos archivos pueden ser leídos por cualquier otro programa mientras se actualizan, pero ningún programa puede actualizar dichos miembros mientras se ejecuta el trabajo. El primer miembro de cada archivo se preasigna con un estado de bloqueo exclusivo con permiso de lectura.

```
//JOB  JOBD(ORDER)
  ALCOBJ  OBJ((FILEA *FILE *EXCLRD) (FILEB *FILE *EXCLRD))
  CALL  PROGX
//ENDJOB
```

Los objetos que están asignados al usuario deben desasignarse en cuando se

termine de utilizarlos, ya que otros usuarios podrían necesitarlos. Sin embargo, los objetos asignados se desasignan automáticamente al final del paso de direccionamiento.

Si los primeros miembros de FILEA y FILEB no se hubiesen preasignado, la restricción exclusiva con permiso de lectura no hubiera entrado en vigor. Al utilizar los archivos, es posible que desee preasignarlos para asegurarse de que no se modifican mientras los utiliza.

Nota: Si un solo objeto se ha asignado más de una vez (mediante más de un mandato de asignación), un único mandato DLCOBJ no desasignará completamente dicho objeto. Se precisa un mandato de desasignación para cada mandato de asignación.

No se produce ningún error si se emite el mandato DLCOBJ para un objeto en el que el usuario no tiene un bloqueo o no tiene el estado de bloqueo cuya asignación se ha solicitado.

Puede cambiar el estado de bloqueo de un objeto, tal como muestra el ejemplo siguiente:

```
  PGM
  ALCOBJ  OBJ((FILEX *FILE *EXCL)) WAIT(0)
  CALL PGMA
  ALCOBJ  OBJ((FILEX *FILE *EXCLRD))
  DLCOBJ  OBJ((FILEX *FILE *EXCL))
  CALL PGMB
  DLCOBJ  OBJ((FILEX *FILE *EXCLRD))
  ENDPGM
```

El archivo FILEX se asigna exclusivamente a PGMA, pero se asigna a PFMB como exclusivo con permiso de lectura.

Puede utilizar bloqueos de registro para asignar registros de datos en un archivo. También puede utilizar el parámetro WAITFILE en un mandato Crear Archivo para especificar cuánto tiempo tiene que esperar el programa a dicho archivo antes de que se produzca una situación de tiempo de espera excedido.

El parámetro WAITRCD en un mandato Crear Archivo especifica cuánto hay que esperar para conseguir un bloqueo del registro. El parámetro DFTWAIT en el mandato Crear Clase (CRTCLS) especifica cuánto tiempo hay que esperar a otros objetos. Para ver una explicación del parámetro WAITRCD, consulte la publicación Backup and Recovery - Advanced.

Para obtener una descripción de las funciones de bloqueo de registro proporcionadas por el control de compromiso, consulte el manual *DB2/400 Programación de la base de datos*.

Subtemas

4.16.1 Visualización de los Estados de Bloqueo para Objetos

Puede utilizar el mandato Trabajar con Bloqueos de Objetos (WRKOBJLCK) o Trabajar con Trabajo (WRKJOB) para visualizar los estados de bloqueo para objetos.

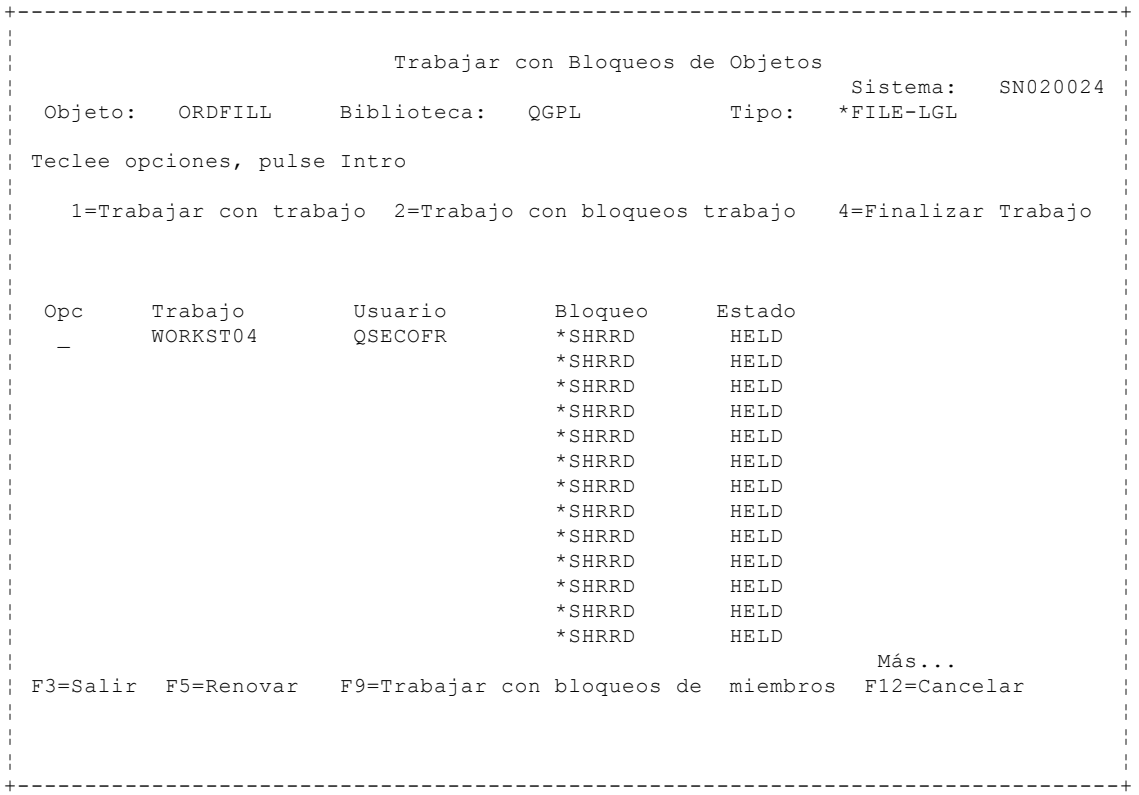
El mandato WRKOBJLCK visualiza todas las peticiones de estado de bloqueo del sistema para un objeto especificado. Visualiza tanto los bloqueos retenidos como los que están a la espera. En el caso de un archivo de base de datos, el mandato WRKOBJLCK visualiza los bloqueos al nivel del archivo (nivel del objeto) pero no al nivel de registro. Por ejemplo, si un archivo de base de datos se abre para actualizarlo, se visualiza el bloqueo del archivo, pero el bloqueo de cualquier registro dentro del archivo no aparece. Los bloqueos de los miembros de archivos de base de datos también se pueden visualizar utilizando el mandato WRKOBJLCK.

Si utiliza el mandato WRKJOB, puede seleccionar la opción de bloqueo del menú Visualizar Trabajo. Esta opción visualiza todas las peticiones de estados de bloqueo pendientes para el trabajo activo especificado, los bloqueos retenidos por el trabajo y los bloqueos para los que está esperando el trabajo. Sin embargo, si un trabajo espera un bloqueo de registros de base de datos, éste no aparece en la pantalla de bloqueos de objeto.

El mandato siguiente visualiza todas las peticiones de estado de bloqueo en el sistema para el archivo lógico ORDFILL:

```
WRKOBJLCK  OBJ(QGPL/ORDFILL)  OBJTYPE(*FILE)
```

La pantalla resultante es:



5.0 Capítulo 5. Trabajar con Objetos en Procedimientos y Programas CL

Las normas para hacer referencia a objetos en mandatos, en programas y en procedimientos CL son las mismas que las que se emplean para hacer referencia a objetos en mandatos que se procesan individualmente (que no forman parte de un procedimiento). Los nombres de objetos pueden estar calificados o no. Siempre que un objeto no está calificado, puede encontrarse mediante una búsqueda en la lista de bibliotecas.

A la mayoría de objetos a los que se hace referencia en programas y procedimientos CL tan sólo se accede en el momento de la ejecución del mandato. Si califica el nombre de un objeto (*biblioteca/nombre*) con una biblioteca específica en la sentencia fuente CL, dicho objeto debe estar en la biblioteca especificada cuando se ejecuta el mandato que le hace referencia, pero no es preciso que el objeto esté en dicha biblioteca cuando se crea el módulo o programa. Esto significa que muchos objetos pueden calificarse en sentencias fuente CL basándose solamente en su ubicación en el momento de la ejecución. Los archivos declarados en un mandato Declarar Archivo (DLCF) y en las definiciones de mandatos son excepciones a esto, porque también se accede a ellas en el momento de la creación. Así pues, cuando se califica uno de estos objetos, debe encontrarse en la biblioteca especificada en el momento de la creación y estar enlazado a dicha biblioteca en el momento de la ejecución. Estas excepciones se tratan en el apartado "Acceso a Objetos en Programas y Procedimientos CL" en el tema 5.1.

Puede evitar esta consideración sobre el tiempo de ejecución para todos los objetos si no se califican los nombres de objetos en las sentencias fuente CL, sino que se hace referencia a la lista de bibliotecas (*LIBL/nombre). Si se hace referencia a esta lista en el momento de la creación, el objeto puede encontrarse en cualquier biblioteca de la lista de bibliotecas en el momento de la ejecución del mandato. Ello es posible siempre que no se tengan objetos con nombres duplicados en distintas bibliotecas. Si utiliza la lista de bibliotecas, puede mover el objeto a una biblioteca distinta entre la creación del procedimiento y el proceso del mandato.

Subtemas

5.1 Acceso a Objetos en Programas y Procedimientos CL

A la mayoría de objetos a los que se hace referencia en programas y procedimientos CL tan sólo se accede hasta que se ejecuta el mandato que les hace referencia. Esto significa que, en el caso de muchos objetos a los que se hace referencia en un programa o procedimiento, el objeto no es necesario que exista cuando se compila el módulo o procedimiento, sino sólo cuando se procesa el mandato concreto del programa (el que hace referencia a ese objeto). Los archivos y definiciones de mandatos son excepciones a esta norma, debido a que se pueden acceder en tiempo de compilación; éstos deben existir antes de que se compile un programa o módulo que les haga referencia. Por esta razón, sus nombres cualificados, si se especifican en el momento de la compilación, deben hacer referencia a su ubicación real en las bibliotecas. Esto no pasa con todos de los objetos y en ningún caso si se utiliza *LIBL.

Puesto que no es preciso que la mayor parte de objetos existan hasta que se ejecuta el programa que les hace referencia, el siguiente programa CL se compila de forma satisfactoria aunque el programa PAYROLL no existe en el momento de efectuarse la compilación:

```
PGM   /* TEST */
DCL ...
MONMSG ...
.
.
.
CALL  PGM(QGPL/PAYROLL)
.
.
.
ENDPGM
```

De hecho, PAYROLL no tiene que existir cuando se llama al procedimiento TEST, sino solamente cuando se procesa el mandato CALL. Así pues, el programa llamado puede crearse en el procedimiento de llamada inmediatamente antes del mandato CALL:

```
PGM   /* TEST */
DCL ...
.
.
.
MONMSG
.
.
.
CRTCLPGM  PGM(QGPL/PAYROLL)
CALL  PGM(QGPL/PAYROLL)
.
.
.
ENDPGM
```

Tenga en cuenta que para los mandatos de creación, como CRTBNDCL o CRTDTAARA, el objeto al que se accede en el momento de la creación o de la ejecución es la definición del mandato de creación, no el objeto que se está creando. Ello significa que si se califica en el tiempo de creación, debe estar en la biblioteca referenciada durante la creación, y en la misma biblioteca que cuando se procesó. Si no se califica, debe encontrarse en alguna de las bibliotecas de la lista de bibliotecas durante la creación y en el tiempo de ejecución. Para evitar este tipo de complicación, no califique los mandatos suministrados por IBM; utilice en su lugar la lista de bibliotecas.

Subtemas

5.1.1 Excepciones: Archivos y Definiciones de Mandatos

Al crear un módulo o programa CL desde sentencias fuente que hacen referencia a procedimientos, definiciones de mandatos o archivos, los objetos deben existir en el momento de la creación, así como cuando se procesa el mandato que hace referencia a los mismos. Ello significa que cuando utiliza el mandato Declarar Archivo (DCLF) debe crear el archivo antes de crear un módulo o programa que haga referencia a dicho archivo. Una vez creado el módulo o programa, puede suprimir el archivo, siempre y cuando exista cuando se procese el mandato que se refiere al mismo.

Subtemas
5.1.1.1 Acceso a Definiciones de Mandatos
5.1.1.2 Acceso a Archivos

5.1.1.1 Acceso a Definiciones de Mandatos

Se accede a las definiciones de mandatos en el momento de la creación y cuando se ejecuta un mandato. El mandato debe existir cuando se crea un módulo o programa que lo utiliza (para permitir la comprobación de sintaxis de sentencias). Si se califica en el momento de la creación, debe encontrarse en la biblioteca a la que se hace referencia durante la creación y cuando se procesa. Si no se califica, debe encontrarse en alguna de las bibliotecas de la lista de bibliotecas durante la creación y en el momento de la ejecución.

Si la definición del mandato pudiera no ser accesible a través de la lista de bibliotecas cuando se ejecuta el programa, o si hay diversas definiciones de mandatos con el mismo nombre, el nombre del mandato debe calificarse en el programa.

El nombre del mandato debe ser el mismo cuando se procesa el programa que cuando se creó el módulo o programa. Si se cambia el nombre de un mandato después de haber creado un módulo o programa que le hace referencia, dicho programa no puede encontrar el mandato cuando se ejecuta y se produce un error. Sin embargo, si se cambia un valor por omisión para un parámetro de un mandato, cuando se procesa ese mandato se utiliza el nuevo valor por omisión. Para más detalles sobre los atributos que pueden modificarse en un mandato sin volver a crearlo, véase el apartado "Consecuencias de Cambiar la Definición del Mandato de un Mandato en un Procedimiento o Programa" en el tema 9.9 y la descripción del mandato Cambiar Mandato (CHGCMD) en la publicación CL Reference.

5.1.1.2 Acceso a Archivos

También se puede acceder a los archivos cuando se compila el mandato que hace referencia a los mismos. Un archivo debe existir antes de crear un programa o módulo CL que lo utilice.

Para crear el archivo, en primer lugar hay que entrar las DDS en un archivo fuente Las DDS describen los formatos de registro y los campos que hay en dichos registros y toda esta información se compila para crear el objeto de archivo mediante el mandato Crear Archivo de Pantalla (CRTDSPF). Los campos descritos en las DDS, deben ser campos de entrada o salida (o ambos), y se declaran al programa o procedimiento CL como variables en el momento de compilar el módulo o programa (exactamente cuando se compila el mandato DCLF del programa). Mediante estas variables, el programa manipula los datos de la pantalla.

Si no se utilizan las DDS para crear un archivo físico, se declara al procedimiento o programa una variable CL para que contenga el registro completo. Esta variable tiene el mismo nombre que el archivo y su longitud es la misma que la longitud de registro del archivo.

Los programas y procedimientos CL no pueden manipular datos en ningún tipo de archivo distinto de archivos de pantalla o archivos de base de datos, excepto con mandatos CL específicos.

Después de la creación del archivo, puede suprimir las DDS, aunque no es aconsejable hacerlo. Una vez compilado el módulo o programa que hace referencia al archivo, puede suprimir también el archivo, siempre y cuando exista, cuando se procesa en el programa el mandato que hace referencia al
|mismo, como Enviar Archivo (SNDF), o Recibir Archivo (RCVF).

Las normas sobre nombres calificados descritas aquí para las áreas de datos y las definiciones de mandatos se aplican también a los archivo. Para obtener más detalles sobre los archivos, véase el apartado "Trabajar con Archivos en Procedimientos CL" en el tema 5.2.

5.1.2 Comprobación de la Existencia de un Objeto

Antes de intentar utilizar un objeto en un programa, puede comprobar si el objeto existe y si dispone de la autorización necesaria para utilizarlo. Esto resulta especialmente útil cuando una función utiliza más de un objeto a la vez.

Para comprobar la existencia de un objeto, utilice el mandato Comprobar Objeto (CHKOBJ). Puede utilizar este mandato en cualquier parte de un procedimiento o programa. El mandato CHKOBJ tiene el siguiente formato:

```
  CHKOBJ  OBJ(nombre-biblioteca/nombre-objeto)  OBJTYPE(tipo-objeto)
```

Otros parámetros opcionales permiten la verificación de la autorización sobre el objeto. Si está comprobando la autorización y tiene la intención de abrir un archivo, debe comprobar la autorización operativa y la de datos.

Cuando se procesa este mandato, los mensajes se envían al programa o procedimiento para informar del resultado de la comprobación del objeto. Puede supervisar estos mensajes y manejarlos como desee. Por ejemplo:

```
             CHKOBJ  OBJ(OELIB/PGMA)  OBJTYPE(*PGM)
             MONMSG  MSGID(CPF9801)  EXEC(GOTO NOTFOUND)
             CALL OELIB/PGMA
             .
             .
             .
  NOTFOUND:  CALL FIX001  /* PGMA Rutina No Encontrada */
             ENDPGM
```

En este ejemplo, el mandato MONMSG comprueba solamente si hay un mensaje de escape de objeto no encontrado. Para obtener una lista de todos los mensajes que el mandato CHKOBJ puede enviar, vea la publicación CL Reference. Puede encontrar información adicional sobre la supervisión de mensajes en el apartado "Utilización del Mandato Supervisar Mensaje (MONMSG)" en el tema 2.5.10 y en el Capítulo 7 y el Capítulo 8.

El mandato CHKOBJ no asigna un objeto. En muchas utilizaciones de la aplicación, la comprobación de la existencia no es una función adecuada y debe realizarse una asignación. El mandato Asignar Objeto (ALCOBJ) proporciona la comprobación de existencia y la asignación.

Puede utilizar el mandato Comprobar Cinta (CHKTAP) o Comprobar Disquete (CHKDKT) en un programa o procedimiento CL para asegurarse de que hay una cinta o disquete en la unidad y que está preparado. Estos mandatos también proporcionan un mensaje de escape que puede supervisar en el programa o procedimiento CL.

5.2 Trabajar con Archivos en Procedimientos CL

En los programas y procedimientos CL se soportan dos tipos de archivos, archivos de pantalla y archivos de base de datos. Puede enviar una pantalla a una estación de trabajo y recibir entrada desde ésta para utilizarla en el programa o procedimiento, o bien puede leer datos de un archivo de bases de datos con la misma finalidad.

Nota: Los archivos de base de datos pueden utilizarse en el programa o procedimiento CL a través de los mandatos DCLF y RCVF. Véase la explicación dada sobre la apertura y cierre de los archivos de base de datos en el manual *DB2/400 Programación de la base de datos*; éste contiene información sobre los mandatos OPNDBF y CLOF, que hacen que los archivos de base de datos estén disponibles para su utilización posterior en programas y procedimientos de lenguaje de alto nivel.

Para utilizar un archivo en un programa o procedimiento CL, debe:

- Dar formato al registro de pantalla o de base de datos, identificando los campos y condiciones que se entran como fuente en las DDS. La utilización de las DDS no es necesaria para un archivo de base de datos.

- Crear el archivo utilizando el mandato Crear Archivo de Pantalla (CRTDSPF), Crear Archivo Físico (CRTPF) o Crear Archivo Lógico (CRTLF). Los programas y procedimientos CL no soportan subarchivos (excepto los de mensajes).

- Para los archivos de base de datos, añadir un miembro al archivo utilizando el mandato Añadir Miembro de Archivo Físico (ADDPFM) o el mandato Añadir Miembro de Archivo Lógico (ADDLFM). Esto no es obligatorio si los mandatos CRTPF o CRTLF añadieron un miembro. El archivo debe tener un miembro cuando se procesa el procedimiento o el
| programa, pero no es necesario que lo tenga cuando se crea este
| procedimiento o programa.

- Hacer referencia al archivo en el procedimiento CL utilizando el mandato DCLF, y hacer referencia al formato de registro en los mandatos CL de manipulación de datos adecuados en el fuente CL.

- Crear el módulo CL.

- Crear el programa.

En un procedimiento CL sólo puede hacerse referencia a un archivo de pantalla o de base de datos. El soporte para los archivos de base de datos y de pantalla es similar, ya que se utilizan los mismos mandatos. Sin embargo, existen algunas diferencias, que se describen a continuación.

- Las sentencias siguientes se aplican solamente a los archivos de pantalla utilizados con los programas y procedimientos CL.

  - Un programa o procedimiento CL sólo puede utilizar los archivos de base de datos con un formato de registro único.

  - El archivo puede ser físico o lógico; un archivo lógico puede estar definido en base a varios miembros de archivos físicos.

  - Solamente se permiten operaciones de entrada con el mandato RCVF. Los parámetros SNDF, SNDRCVF, ENDRCV, WAIT y DEV en el mandato RCVF no están permitidos para archivos de base de datos.

  - Las DDS no son necesarias para crear un archivo físico al que se hace referencia en un programa o procedimiento CL. Si las DDS no se utilizan para crear un archivo físico, el archivo tiene un formato de registro con el mismo nombre que el archivo y hay un sólo campo en el formato de registro con el mismo nombre que el archivo y con la misma longitud que la longitud de registro del archivo (parámetro RCDLEN del mandato CRTPF).

  | - El archivo no necesita tener un miembro cuando se crea para el
  | módulo o programa. Sin embargo, debe tener un miembro cuando se
  | procesa por el programa.

  | - El archivo se abre solamente para recibir entrada cuando se
  | procesa el primer mandato RCVF. El archivo debe existir y tener
  | un miembro.

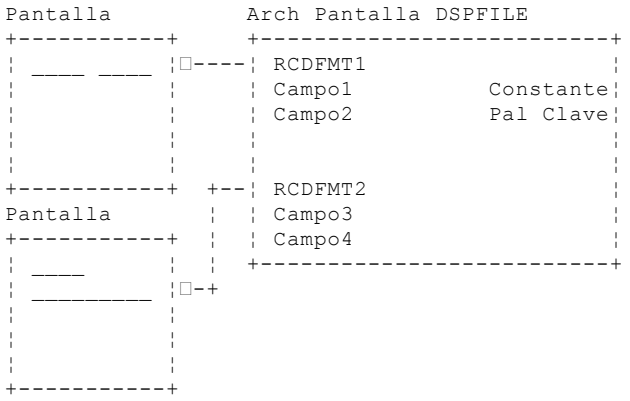
  - El archivo permanece abierto hasta que el procedimiento o programa OPM retorna o se alcanza el final del archivo. Cuando se alcanza el final del archivo, se envía un mensaje CPF0864 al programa o

procedimiento CL, y no se permiten operaciones adicionales para el archivo. El procedimiento o programa debe supervisar este mensaje y efectuar la acción adecuada cuando se alcance el final del archivo.

- Las sentencias siguientes se aplican sólo a archivos de pantalla utilizados con programas y procedimientos CL:
  - Los archivos de pantalla pueden tener un máximo de 99 formatos de registro.
  - Los archivos de pantalla permiten todos los mandatos de manipulación de datos (SNDF, SNDRCVF, RCVF, ENDRCV y WAIT).
  - El archivo de pantalla debe definirse con las DDS.
  - Se abre el archivo de pantalla para entrada y salida cuando se procesa el primer mandato SNDF, SNDRCVF o RCVF. El archivo permanece abierto hasta que retorna el procedimiento o programa OPM.

Nota: La apertura no se produce para ambos tipos de archivos hasta que se produce el primer envío o recepción. Por ello, puede crearse durante el programa o procedimiento el archivo que se va a utilizar y puede realizarse una alteración temporal antes del primer envío o recepción.

El formato para la pantalla se identifica como un formato de registro en las DDS. Cada formato de registro puede contener campos (entrada, salida y entrada/salida), condiciones/indicadores y constantes. Pueden entrarse varios formatos de registro en un archivo de pantalla. Los nombres de archivo de pantalla, de formato de registro y de campo deben ser exclusivos, porque puede ser un requisito de otros lenguajes de alto nivel HLL, aunque los programas o procedimientos CL no lo necesiten.



Consulte el manual DDS Reference para obtener más información acerca de las DDS.

Puede utilizar los métodos descritos en la publicación Application Display Programming o en la Ayuda para el Diseño de Pantallas (SDA) para entrar el fuente de las DDS para registros y campos en el archivo de pantalla. Consulte el manual *ADTS/400: Ayuda para el Diseño de Pantallas (SDA)* para ver información detallada referente al diseño interactivo de pantallas.

Un programa o procedimiento CL puede utilizar diversos mandatos, llamados mandatos de manipulación de datos. Estos mandatos le permiten hacer referencia a un archivo de pantalla para enviar y recibir datos de los dispositivo de pantalla. Estos mandatos también le permiten hacer referencia a un archivo de base de datos para leer registros desde un archivo de base de datos. Estos mandatos son los siguientes:

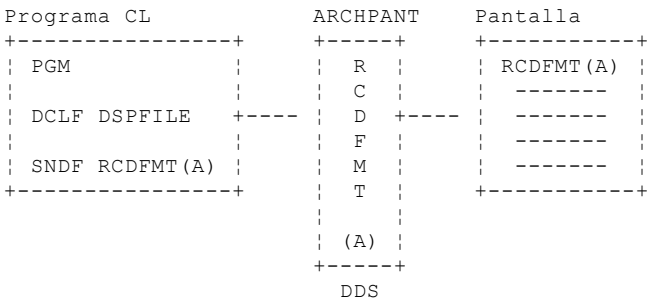
- Declarar Archivo (DCLF). Define un archivo de pantalla o de base de datos que se va a utilizar en un programa o procedimiento. Los campos del archivo se declaran automáticamente como variables para su uso en el programa o procedimiento.
- Enviar Archivo (SNDF). Envía datos a la pantalla.
- Recibir Archivo (RCVF). Recibe datos de la pantalla o de la base de datos.

- Enviar/Recibir Archivo (SNDRCVF). Envía datos a la pantalla; a continuación solicita entrada y, de forma opcional, recibe datos de la pantalla.

- Alterar Temporalmente con Archivo de Pantalla (OVRDSPF). Alterar temporalmente en tiempo de ejecución, un archivo utilizado por un procedimiento o programa con un archivo de pantalla.

- Alterar Temporalmente con Archivo de Base de Datos (OVRDBF). Alterar temporalmente en tiempo de ejecución, un archivo utilizado por un procedimiento o programa con un archivo de base de datos.

Estos mandatos permiten que un programa en ejecución se comunique con un dispositivo de pantalla utilizando las funciones de pantalla proporcionadas por las DDS, y leer registros de un archivo de base de datos. Las DDS suministran funciones para escribir menús y realizar peticiones básicas de datos orientados a aplicaciones que son características de muchas aplicaciones CL.



Los campos de la pantalla o del registro se identifican en las DDS para el archivo. Para que el programa o procedimiento CL utilice los campos, el mandato DCLF debe hacer referencia al archivo en el programa o procedimiento CL. Esta referencia motiva que los campos e indicadores del archivo se declaren automáticamente en su programa o procedimiento como variables. Puede utilizar estas variables en los mandatos CL del modo que desee; sin embargo, su finalidad principal es enviar y recibir información de la pantalla. El mandato DCLF no se utiliza durante la ejecución.

El formato de la pantalla y las opciones para los campos se especifican en el archivo de dispositivo y se controlan mediante la utilización de indicadores. Puede utilizar hasta 99 valores de indicador con las DDS y el soporte CL. Las variables de indicador se declaran en el programa o procedimiento CL como variables lógicas, con nombres que van desde &IN01 a &IN99, para cada indicador que aparezca en los formatos de registro del archivo de dispositivo al que se hace referencia en el mandato DCLF. Los indicadores permiten visualizar los campos y controlar las funciones de visualización de gestión de datos, así como proporcionar información de respuesta al programa o procedimiento desde el dispositivo de pantalla. Los indicadores no se utilizan con archivos de base de datos.

Subtemas

5.2.1 Referencia a Archivos en un Procedimiento CL

Se accede a los archivos durante la compilación de los mandatos DCLF cuando se crean los módulos y programas CL de modo que pueden declararse variables para cada campo del archivo.

Si ha calificado el archivo en la compilación, dicho archivo debe estar en la biblioteca en el momento de la ejecución. Si ha utilizado la lista de bibliotecas en el momento de la compilación, el archivo debe estar en una biblioteca de la lista de bibliotecas en el momento de la ejecución.

5.2.2 Apertura y Cierre de Archivos en un Procedimiento CL

Cuando utiliza el soporte CL, puede hacerse referencia a un solo archivo de un procedimiento o programa OPM. Este archivo se abre automáticamente cuando se lleva a cabo la primera operación de envío, recepción o envío/recepción. Un archivo de pantalla abierto permanece abierto hasta que el procedimiento o programa OPM en el que se ha abierto retorna o transfiere el control. Un archivo de base de datos abierto se cierra cuando se alcanza el final del archivo, o cuando el procedimiento o programa OPM en el que se ha abierto devuelve o transfiere el control. Una vez se ha cerrado un archivo de base de datos, no se puede volver a abrir en la misma llamada del procedimiento o programa OPM.

Cuando se abre un archivo de base de datos, se abre el primer miembro del mismo, a menos que se haya utilizado previamente un mandato OVRDBF para especificar otro miembro (parámetro MBR). Si un procedimiento o programa OPM finaliza debido a un error, se cierran los archivos. Puesto que un archivo permanece abierto hasta que el procedimiento o programa OPM en el que se ha abierto finalice el proceso, ésta es una manera sencilla de que los procedimientos y los programas en ejecución compartan vías abiertas de datos. Se puede abrir un archivo en un procedimiento o programa y posteriormente su vía abierta de datos se puede compartir con otro programa o procedimiento, bajo una de las siguientes condiciones:

- El archivo se creó con el atributo SHARE(*YES) o se ha modificado para que lo tenga.
- Está en vigor una alteración temporal para el archivo en el que se especifica SHARE(*YES).

Los archivos pueden compartirse de esta forma entre dos procedimientos o programas. Para obtener una descripción detallada de la función disponible cuando se comparten vías abiertas de datos, vea la descripción del parámetro SHARE en los mandatos CRTDSPF, CRTPF y CRTLF en el manual CL Reference. Cuando se abre un archivo de pantalla en un procedimiento CL o programa OPM, siempre se abre tanto para la entrada como para la salida. Cuando se abre un archivo de base de datos en un programa o procedimiento CL, se abre sólo para entrada.

No debe especificar LVL(*CALLER) en el mandato Reclamar Recursos (RCLRSC) en los programas y procedimientos CL que utilizan archivos. De lo contrario, todos los archivos abiertos por el procedimiento o programa OPM se cerrarán inmediatamente, y cualquier intento de acceder al archivo finalizaría de forma anómala.

5.2.3 Declaración de un Archivo

El mandato Declarar Archivo (DCLF) se utiliza para declarar un archivo de pantalla o de base de datos en su programa o procedimiento CL. El mandato DCLF no se puede utilizar para declarar archivos, tales como archivos de cinta, de disquete, de impresora y mixtos. Sólo se permite un mandato DCLF en un procedimiento CL o programa OPM. El mandato DCLF tiene los siguientes parámetros:

```
DCLF   FILE(nombre-biblioteca/nombre-archivo)
       RCDFMT(nombres-formato-registro)
```

Observe que el archivo debe existir antes compilar el módulo o programa.

Si utiliza un archivo en su procedimiento o programa, puede que tenga que especificar campos de entrada/salida en las DDS. Estos campos se manejan como variables en el procedimiento o programa. Cuando se utiliza un mandato DCLF, el compilador CL declara las variables CL para cada campo e indicador de opción en cada formato de registro del archivo. Para un campo, el nombre de una variable CL es el nombre del campo precedido de un símbolo &. Para un indicador de opción, el nombre de la variable CL es el indicador precedido de &IN.

Por ejemplo, si en las DDS hay definidos un campo llamado INPUT y un indicador 10, el mandato DCLF los declara automáticamente como &INPUT e &IN10. Esta declaración se realiza cuando se compila el módulo o programa CL. En un mandato pueden especificarse 50 nombres de formato de registro como máximo, pero ninguno de ellos puede ser una variable. Para un archivo de base de datos, sólo puede especificarse un formato de registro.

Si se utilizaron las siguientes DDS para crear un archivo de pantalla denominado CNTRLDPS en la biblioteca MCGANN:

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R MASTER
     A                                       CA01(01 'F1 RESPUESTA')
     A            TEXT         300      2  4
     A            RESPONSE      15   1  8  4 BLINK
     A
     A
```

Habrán tres variables, &IN01, &TEXT y &RESPONSE, disponibles desde el archivo de pantalla. En un procedimiento CL que haga referencia a este archivo de pantalla, el usuario entraría solo la sentencia fuente DCLF:

```
DCLF MCGANN/CNTRLDSP
```

El compilador ampliará esta sentencia para declarar individualmente todas las variables del archivo de pantalla. La declaración ampliada en la lista del compilador tiene el siguiente formato:

```
    500-               DCLF        MCGANN/CNTRLDSP
          QUALIFIED FILE NAME    'MCGANN    '  'CNTRLDSP   '

            RECORD FORMAT NAME   'MASTER    '


              CL VARIABLE     TYPE     LENGTH     PRECISION (IF *DEC)
               &IN01          *LGL        1
               &TEXT          *CHAR     300
               &RESPONSE      *CHAR      15
```

5.2.4 Envío y Recepción de Datos con un Archivo de Pantalla

Los únicos mandatos que se pueden utilizar con un archivo de pantalla para enviar o recibir datos en procedimientos o programas CL son los mandatos SNDF, RCVF y SNDRCVF.

Cuando se ejecuta un mandato SNDF, el sistema da formato al contenido de las variables asociadas con los campos de salida o de salida/entrada del formato de registro especificado y lo envía al dispositivo de pantalla. De la misma forma, cuando se ejecuta un mandato RCVF, los valores de los campos asociados a los campos de entrada o de salida/entrada del formato de registro de la pantalla se colocan en las variables CL correspondientes.

El mandato SNDRCVF envía el contenido de las variables CL a la pantalla y después realiza una operación equivalente a la del mandato RCVF para obtener los campos actualizados de la pantalla. Observe que el lenguaje CL no da soporte a los números decimales con zona. Por lo tanto, los campos del archivo de pantalla definidos como decimales con zona, hacen que se definan campos *DEC en el programa o procedimiento CL. Se da soporte a los campos *DEC internamente como decimales empaquetados, y los mandatos CL convierten los tipos de datos empaquetados y con zona cuando es necesario. Los campos que se solapan en el archivo de pantalla debido a que hay posiciones de pantalla que coinciden dan como resultado variables CL definidas por separado que no se solapan. Los formatos de registro que contiene datos de coma flotante no se pueden utilizar en un programa o procedimiento CL.

Nota: Si un mandato SNDRCVF o RCVF para una estación de trabajo indica WAIT(*NO), o si se emite un mandato SNDF utilizando un formato de registro que contiene la palabra clave INVITE DDS, el mandato WAIT se utiliza para recibir datos.

Excepto para los subarchivos de mensajes, cualquier intento de enviar o recibir registros de subarchivos causa errores en la ejecución. La mayoría de las demás funciones especificadas para los archivos de pantalla en las DDS están disponibles; algunas de ellas (como la utilización de números de línea inicial variables) no lo están. Para obtener más información acerca de mensajes y subarchivos en programas y procedimientos CL, consulte el Capítulo 8.

El ejemplo siguiente muestra los pasos necesarios para crear un menú de operador y enviar y recibir datos utilizando el mandato SNDRCVF. El menú aparece del siguiente modo:

```
+-------------------------------------------------------------------------------+
|                                                                               |
| Menú operador                                                                 |
|                                                                               |
|    1. Cuentas a pagar                                                         |
|    2. Cuentas a cobrar                                                        |
|    90. Acabar                                                                 |
|                                                                               |
|  Opción:                                                                      |
|                                                                               |
|                                                                               |
|                                                                               |
+-------------------------------------------------------------------------------+
```

En primer lugar, entre la siguiente fuente DDS. El formato de registro es MENU y OPTION es un campo con posibilidad de entrada. El campo OPTION utiliza DSPATR(MDT). Ello hace que el sistema compruebe si los valores de este campo son válidos aunque el operador no entre nada.

```
  |...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
       A          R MENU
       A                                    1  2'Menú operador'
       A                                    3  4'1. Cuentas a pagar'
       A                                    5  4'2. Cuentas a cobrar'
       A                                    5  4'90. Acabar'
       A                                    7  2'Opción'
       A            OPTION         2Y 01   + 2VALUES(1 2 90) DSPATR(MDT)
       A
       A
```

Entre el mandato CRTDSPF para crear el archivo de pantalla. En programación CL, el nombre del archivo de pantalla (INTMENU) puede ser el mismo que el nombre de formato de registro (MENU), aunque ello no es válido para otros lenguajes, como RPG para OS/400.

El archivo de pantalla puede crearse también utilizando el programa de

utilidad de Ayuda para el Diseño de Pantallas (SDA).

A continuación, se entra el fuente CL para ejecutar el menú.

El fuente CL para este menú es el siguiente:

```
            PGM /* MENÚ OPERADOR */
            DCLF INTMENU
  BEGIN:    SNDRCVF RCDFMT(MENU)
            IF COND(&OPTION *EQ 1) THEN(CALL ACTSPAYMNU)
            IF COND(&OPTION *EQ 2) THEN(CALL ACTSRCVMNU)
            IF COND(&OPTION *EQ 90) THEN(SIGNOFF)
            GOTO BEGIN
            ENDPGM
```

Cuando se compila este fuente, el mandato DCLF declara automáticamente el campo de entrada OPTION en el procedimiento como una variable.

El mandato SNDRCVF toma el valor por omisión de WAIT(*YES); es decir, el programa espera hasta que el programa recibe la entrada.

5.2.5 Creación de un Programa CL para Controlar un Menú

El siguiente ejemplo muestra cómo puede escribirse un procedimiento CL para visualizar y controlar un menú. Consulte el manual Application Display Programming para ver otra forma de crear y controlar menús.

Este ejemplo muestra un procedimiento CL, ORD040C, que controla la visualización del menú general del departamento de pedidos y determina a qué procedimiento HLL hay que llamar según la opción seleccionada en el menú. El procedimiento muestra el menú en la estación de pantalla.

El menú general del departamento de pedidos tiene este aspecto:

```
+-------------------------------------------------------------------------------+
|                                                                               |
|  Menú General Dpto. Pedidos                                                   |
|                                                                               |
|   1 Consultar archivo clientes                                                |
|   2 Consultar archivo artículos                                               |
|   3 Búsqueda por nombre cliente                                               |
|   4 Consultar pedidos de un cliente                                           |
|   5 Consultar pedido existente                                                |
|   6 Entrada de pedidos                                                        |
|  98 Fin de menú                                                               |
|                                                                               |
|  Opción:                                                                      |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
+-------------------------------------------------------------------------------+
```

Las DDS para el archivo de pantalla ORD040C es como el siguiente:

```
  |...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
       A* MENU ORDO4OCD ORDER DEPT GENERAL MENU
       A
       A          R MENU                       TEXT('Menú General')
       A                                    1  2'Menú General Dpto. Pedidos'
       A                                    3  3'1 Consultar archivo clientes'
       A                                    4  3'2 Consultar archivo artículos'
       A                                    5  3'3 Búsqueda por nombre cliente'
       A                                    6  3'4 Consultar pedidos de un +
       A                                        cliente'
       A                                    7  3'5 Consultar pedido existente'
       A                                    8  3'6 Entrada de pedidos'
       A                                    9  2'98 Fin de menú'
       A                                   11  2'Opción'
       A            RESP           2Y001 11 10VALUES(1 2 3 4 5 6 98)
       A                                       DSPATR(MDT)
       A
       A
```

El procedimiento fuente para ORD040C es de la siguiente manera:

```
         PGM /* ORD040C Menú General Dpto. Pedidos */
         DCLF FILE(ORD040CD)
  START: SNDRCVF RCDFMT(MENU)
         IF (&RESP=1) THEN(CALLPRC CUS210)
         /*Consulta cliente */
         ELSE +
           IF (&RESP=2) THEN(CALLPRC ITM210)
           /**Consulta artículo*/
           ELSE +
             IF (&RESP=3) THEN(CALLPRC CUS220)
             /* Búsqueda nombre cliente */
             ELSE +
               IF (&RESP=4) THEN(CALLPRC ORD215)
               /* Pedidos por cliente */
               ELSE +
                 IF (&RESP=5) THEN(CALLPRC ORD220)
```

```
            /* Pedido existente */
            ELSE +
              IF (&RESP=6) THEN(CALLPRC ORD410C)
              /* Entrada pedidos */
              ELSE +
                IF (&RESP=98) THEN(RETURN)
                /* Fin de Menú */
      GOTO START
      ENDPGM
```

El mandato DCLF indica qué archivo contiene los atributos de campo que el sistema precisa para dar formato al menú general del departamento de pedidos cuando se procesa el mandato SNDRCVF. El sistema declara automáticamente una variable para cada campo del formato de registro del archivo especificado si dicho formato se utiliza en un mandato SNDF, RCVF o SNDRCVF. El nombre de la variable para cada campo declarado automáticamente es un símbolo & seguido del nombre del campo. Por ejemplo, el nombre de la variable del campo de respuesta RESP de ORD040C es &RESP.

A continuación se facilitan otras observaciones sobre el funcionamiento de este menú:

El mandato SNDRCVF se utiliza para enviar el menú a la pantalla y para recibir la opción seleccionada en la pantalla.

Si la opción seleccionada en el menú es **98**, ORD040C vuelve al procedimiento que lo llamó.

Las sentencias ELSE son necesarias para procesar las respuestas como alternativas que se excluyen mutuamente.

Nota: Este menú se ejecuta utilizando el mandato CALL. Consulte el manual Application Display Programming para ver una explicación de los menús que se ejecutan utilizando el mandato GO.

5.2.6 Alteración Temporal de Archivos de Pantalla en un Procedimiento CL

Puede utilizar el mandato Alterar Temporalmente con Archivo de Pantalla (OVRDSPF) para sustituir el archivo de pantalla especificado en un programa o procedimiento CL, o para cambiar determinados parámetros del archivo de pantalla existente. Esto puede resultar especialmente útil para aquellos archivos que se han redenominado o movido desde que se compiló el módulo o programa.

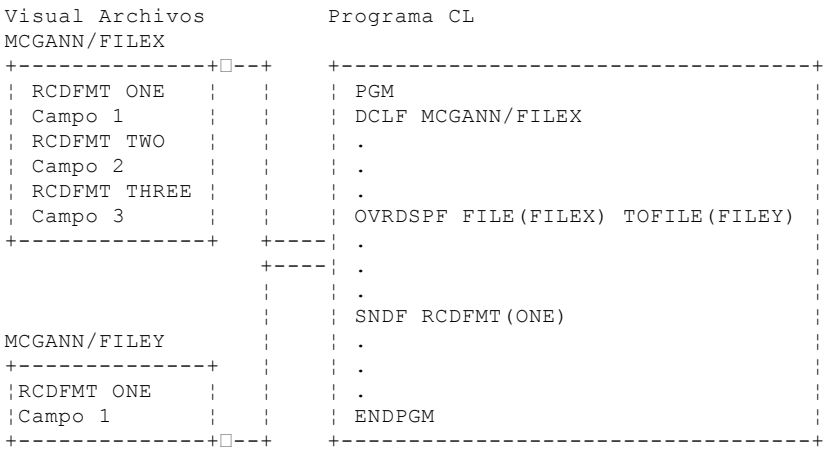
Los parámetros iniciales del mandato OVRDSPF son:

```
  OVRDSPF   FILE(nombre-archivo-alterado-temporalmente) TOFILE(nombre-archivo-nuevo)
            DEV(nombre-dispositivo)
```

El mandato OVRDSPF es válido para un archivo al que un programa o procedimiento CL hace referencia, solamente si el archivo especificado en el mandato DCLF era un archivo de pantalla al crear el módulo o programa. El archivo utilizado cuando se ejecuta el programa debe ser del mismo tipo que el archivo al que se hizo referencia cuando se creó el módulo o programa.

El mandato OVRDSPF debe ejecutarse antes de que el archivo que va a ser alterado temporalmente se abra para su utilización. La primera utilización de un mandato de envío o recepción produce una apertura. El archivo se altera temporalmente si se abre en el procedimiento o programa que contiene el mandato OVRDSPF, si se abre en otro procedimiento al que el mandato CALLPRC transfiere el control, o si se abre en otro programa al que el mandato CALL transfiere el control. Consulte la publicación *Gestión de datos* para obtener más información acerca de cómo alterar temporalmente los archivos.

Cuando se altera temporalmente un archivo con otro, los únicos nombres de formato de registro que han de estar en el archivo de alteración temporal son los nombres a los que se hace referencia en los mandatos SNDF, RCVF o SNDRCVF. En la ilustración siguiente, el archivo de pantalla FILEY no necesita el formato de registro TWO ni THREE.



Debe asegurarse de que el formato de registro hace referencia a nombres del archivo original y de que los archivos de alteración temporal tienen los mismos nombres de indicadores y definiciones de campo en el mismo orden. Puede obtener resultados inesperados si especifica LVLCHK(*NO).

Otra consideración se refiere al parámetro DEV en los mandatos SNDF, RCVF y SNDRCVF cuando se aplica un mandato OVRDSPF. Si se especifica *FILE en el parámetro DEV de los mandatos RCVF, SNDF o SNDRCVF, el sistema dirige automáticamente la operación al dispositivo correcto para el archivo alterado temporalmente. Si especifica un dispositivo específico en la palabra clave DEV de los mandatos RCVF, SNDF o SNDRCVF, puede darse una de estas situaciones:

- Si está utilizando un archivo de dispositivo de pantalla único, se producirá un error si el archivo de pantalla se altera temporalmente a un dispositivo distinto del especificado en el mandato RCVF, SNDF o SNDRCVF.
- Si está utilizando un archivo de múltiples dispositivos de pantalla, se producirá un error si el dispositivo especificado en el mandato RCVF, SNDF o SNDRCVF no se encuentra entre los especificados en el mandato OVRDSPF.

5.2.7 Trabajo con Archivos de Pantalla de Múltiples Dispositivos

La modalidad normal de operación en un sistema es que el usuario de la estación de trabajo se conecte y se convierta en el solicitante de un trabajo interactivo. Muchos usuarios pueden hacerlo al mismo tiempo, ya que cada uno utiliza una copia lógica del procedimiento, incluyendo el archivo de pantalla del procedimiento. Cada solicitante llama a un trabajo distinto en esta modalidad. Esto no se considera un uso de múltiples dispositivos de pantalla.

Se produce una configuración de múltiples dispositivos de pantalla cuando un solo trabajo llamado desde un peticionario se comunica con varias estaciones de pantalla mediante un archivo de pantalla. Mientras que un procedimiento CL sólo puede manejar un archivo de pantalla, el archivo de pantalla o diferentes formatos de registro del mismo, pueden enviarse a diversos dispositivos de pantalla. Los mandatos más utilizados con los archivos de múltiples dispositivos de pantalla son los siguientes:

- Finalizar Recepción (ENDRCV). Este mandato cancela las peticiones de entrada que no se han satisfecho.
- Esperar (WAIT). Acepta la entrada desde cualquier dispositivo de pantalla desde el cual solicitaron datos de usuario uno o varios mandatos RCVF o SNDRCVF previos cuando en el mandato estaba especificado WAIT(*NO), o uno o más mandatos previos SNDF a un formato de registro que contiene la palabra clave INVITE DDS.

Si utiliza un archivo de múltiples dispositivos de pantalla, los nombres de dispositivo deben estar especificados en el parámetro DEV del mandato CRTDSPF cuando se crea el archivo de pantalla, en el mandato CHGDSPF cuando se modifica el archivo de pantalla o en un mandato de alteración temporal, y el número de dispositivos debe ser inferior o igual al número especificado en el parámetro MAXDEV del mandato CRTDSPF.

Las configuraciones de múltiples dispositivos de pantalla afectan a los mandatos SNDRCVF y RCVF y puede ser necesaria la utilización de los mandatos WAIT o ENDRCV. Cuando se utiliza un mandato RCVF o SNDRCVF con múltiples dispositivos de pantalla, el valor por omisión WAIT(*YES) impide el proceso posterior hasta que un campo con posibilidad de entrada se devuelva al programa desde el dispositivo nombrado en el parámetro DEV. Dado que la respuesta puede demorarse, a veces es útil especificar WAIT(*NO), y dejar así que el procedimiento o programa siga ejecutando otros mandatos antes de satisfacer la operación de recepción.

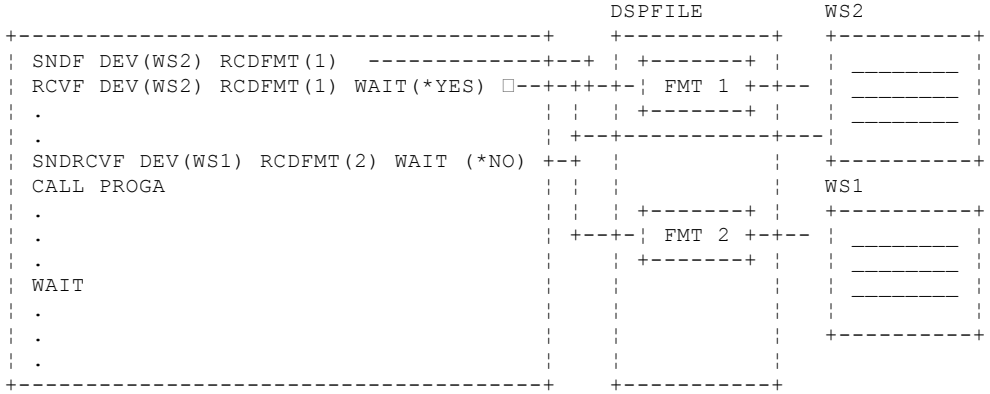
Si utiliza un mandato RCVF o SNDRCVF y especifica WAIT(*NO), el programa o procedimiento CL sigue ejecutandose hasta que se procese un mandato WAIT.

La utilización de un mandato SNDF con un formato de registro que tenga la palabra clave DDS INVITE es equivalente al uso de un mandato SNDRCVF con WAIT(*NO) especificado. La palabra clave DDS INVITE se pasa por alto para los mandatos SNDRCVF y RCVF.

El mandato WAIT debe emitirse para acceder a un registro de datos. Si no hay datos disponibles, el proceso se suspende hasta que se reciben datos desde un dispositivo de pantalla, o hasta que se ha sobrepasado el tiempo límite especificado en el parámetro WAITRCD para el archivo de pantalla en los mandatos CRTDSPF, CHGDSPF o OVRDSPF. Si se sobrepasa el límite de tiempo, se emite el mensaje CPF0889.

El mandato WAIT también puede satisfacerse mediante la cancelación del trabajo con la opción controlada en los mandatos ENDJOB, ENDSYS, PWRDWNSYS y ENDSBS. En este caso, se emite el mensaje CPF0888 y no se devuelve ningún dato. Si se emite un mandato WAIT sin una petición de recepción anterior (como RCVF . . . WAIT(*NO)), se produce un error de proceso.

Una configuración típica de múltiples dispositivos de pantalla (codificada) sería la siguiente:



En el ejemplo anterior, los dos primeros mandatos muestran una secuencia típica en la que se toma el valor por omisión; el proceso espera a que la operación de recepción desde WS2 finalice. Dado que en el parámetro DEV está especificado WS2, el mandato RCVF no se ejecuta hasta que WS2 responde, aunque se satisfagan las peticiones anteriores pendientes (no mostradas) de otras estaciones.

Sin embargo, el mandato SNDRCVF tiene WAIT(*NO) especificado y, por tanto, no espera una respuesta de WS1. En cambio, prosigue el proceso y se llama a PROGA. El proceso entonces se detiene en el mandato WAIT hasta que una estación de trabajo satisfaga una petición pendiente o hasta que la función alcance el tiempo de espera excedido.

El mandato WAIT tiene el formato siguiente:

```
  WAIT DEV(nombre-variable-CL)
```

Si especifica el parámetro DEV, el nombre de la variable CL es el nombre del dispositivo que respondió. (El valor por omisión es *NONE.) Si hay varias peticiones de recepción (tales como RCVF. . . WAIT(*NO)), esta variable toma el nombre del primer dispositivo que responde después de llegar al mandato WAIT, y el proceso continúa. Los datos recibidos se colocan en la variable asociada con el campo del dispositivo de pantalla.

Puede utilizar un mandato RCVF con WAIT(*YES) especificado para esperar los datos de un dispositivo específico. Debe especificarse el mismo nombre de formato de registro tanto para la operación que arrancó la petición de recepción como para el mandato RCVF.

En algunos casos, hay varias peticiones de recepción pendientes, pero el proceso no puede seguir sin una respuesta por parte de un dispositivo de pantalla determinado. En el ejemplo siguiente, tres mandatos especifican WAIT(*NO), pero el proceso no puede continuar en la etiqueta LOOP hasta que WS3 responda:

```
           PGM
           .
           .
           .
           SNDF DEV(WS1) RCDFMT(ONE)
           SNDF DEV(WS2) RCDFMT(TWO)
           SNDRCVF DEV(WS3) RCDFMT(THREE) WAIT(*NO)
           RCVF DEV(WS2) RCDFMT(TWO) WAIT(*NO)
           RCVF DEV(WS1) RCDFMT(ONE) WAIT(*NO)
           CALL...
           CALL...
           .
           .
           RCVF DEV(WS3) RCDFMT(THREE) WAIT(*YES)
  LOOP:    WAIT DEV(&WSNAME)
           MONMSG CPF0882 EXEC(GOTO REPLY)
           .
           .
           .
           GOTO LOOP
  REPLY:   CALL...
           .
           .
           .
           ENDPGM
```

Los programas y procedimientos CL también soportan el mandato ENDRCV, que le permite cancelar una petición para entrada que todavía no se ha realizado. Un mandato SNDF o SNDRCVF también le permite llevar a cabo esta misma acción. Sin embargo, si los datos estaban disponibles en el momento en que se procesó el mandato SNDF o SNDRCVF, se enviará el mensaje CPF0887. En tal caso, los datos deben recibirse con el mandato WAIT o RCVF, o bien deberá cancelarse explícitamente la petición con un mandato ENDRCV para poder procesar el mandato SNDF o SNDRCVF.

5.2.8 Recepción de Datos desde un Archivo de Base de Datos

El único mandato que puede utilizarse para recibir entrada de un archivo de base de datos es RCVF.

Cuando se ejecuta un mandato RCVF, se lee el siguiente registro de la vía de acceso del archivo y los valores de los campos definidos en el formato de registro de base de datos se colocan en las variables CL correspondientes. Observe que el lenguaje CL no da soporte a los números binarios ni decimales con zona. Por lo tanto, los campos del archivo de base de datos definidos como decimales con zona o binarios se definen campos *DEC en el programa o procedimiento CL. Se da soporte a los campos *DEC internamente como decimales empaquetados, y el mandato RCVF convierte los datos binarios y decimales con zona a decimales empaquetados cuando es necesario. Los archivos de base de datos que contienen datos de coma flotante no se pueden utilizar en un procedimiento o programa CL.

Cuando se llega al final del archivo, se envía el mensaje CPF0864 al procedimiento o programa OPM. El proceso del mandato RCVF no modifica las variables CL declaradas para el formato de registro cuando se envía dicho mensaje. Es necesario controlar si aparece este mensaje y realizar la acción adecuada para el final del archivo. Si intenta ejecutar más mandatos RCVF después de llegar al final del archivo, se vuelve a enviar el mensaje CPF0864.

5.2.9 Alteración Temporal de Archivos de Base de Datos en un Programa o Procedimiento CL

Puede utilizar el mandato Alterar Temporalmente con Archivo de Base de Datos (OVRDBF) para sustituir el archivo de base de datos especificado en un procedimiento o programa CL o para cambiar determinados parámetros del archivo de base de datos existente. Esto puede resultar especialmente útil para aquellos archivos que se han redenominado o movido desde que se creó el procedimiento o el programa. También se puede utilizar para acceder a un miembro de archivo que no sea el primer miembro.

Los parámetros iniciales del mandato OVRDBF son:

```
  OVRDBF FILE(nombre-archivo-alterado-temporalmente) TOFILE(nombre-archivo-nuevo)
         MBR(nombre-miembro)
```

El mandato OVRDBF es válido para un archivo al que un procedimiento o programa CL hace referencia, sólo si el archivo especificado en el mandato DCLF era un archivo de base de datos cuando se creó el módulo o programa. El archivo utilizado al procesar el programa, debe ser del mismo tipo que el archivo al que se hizo referencia cuando se creó el módulo o programa.

El mandato OVRDBF debe procesarse antes de que se abra para su utilización el archivo que va a alterarse temporalmente (la primera utilización del mandato RCVF produce una apertura). El archivo se altera temporalmente si se abre en el procedimiento o programa OPM que contiene el mandato OVRDBF, si se abre en otro programa al que se ha transferido el control mediante el mandato CALL, o si se abre en otro procedimiento al que se ha transferido el control utilizando el mandato CALLPRC. Consulte el manual *Gestión de datos* para obtener más información referente al mandato OVRDBF.

Cuando se altera temporalmente un archivo con otro, el archivo de alteración temporal debe tener un solo formato de registro. Puede utilizarse un archivo lógico que tenga varios formatos de registro definidos en las DDS si está definido basándose en un solo miembro de archivo físico. Un archivo lógico que tenga solamente un formato de registro definido en las DDS puede estar definido basándose en más de un miembro de archivo físico. No es necesario que el nombre del formato sea el mismo que el nombre del formato al que se hizo referencia cuando se creó el programa. Deberá tener la seguridad de que el formato de los datos en el archivo que altera temporalmente es el mismo que en el archivo original. Puede obtener resultados inesperados si especifica LVLCHK(*NO). Consulte el manual *DB2/400 Programación de la base de datos* para obtener más información acerca de las consideraciones relativas a LVLCHK.

5.2.10 Referencia a Archivos de Salida desde Mandatos de Pantalla

Varios mandatos de visualización de IBM le permiten colocar la salida del mandato en un archivo de base de datos (parámetro OUTFILE). Los datos de este archivo pueden recibirse y procesarse directamente en un procedimiento o programa CL. Consulte la publicación *DB2/400 Programación de la base de datos* para ver una explicación de algunos de estos mandatos y las descripciones de formato.

El siguiente procedimiento CL acepta dos parámetros, el nombre de usuario y un nombre de biblioteca. El procedimiento determina los nombres de todos los programas, archivos y áreas de datos de la biblioteca y otorga autorización normal a los usuarios especificados.

```
       PGM PARM(&USER &LIB)
       DCL &USER *CHAR 10
       DCL &LIB *CHAR 10
  (1)  DCLF QSYS/QADSPOBJ
  (2)  DSPOBJD OBJ(&LIB/*ALL) OBJTYPE(*FILE *PGM *DTAARA) +
       OUTPUT(*OUTFILE) OUTFILE(QTEMP/DSPOBJD)
  (3)  OVRDBF QADSPOBJ TOFILE(QTEMP/DSPOBJD)
  (4)  READ:  RCVF
  (5)  MONMSG CPF0864 EXEC(RETURN)  /* SALIR AL LLEGAR AL FINAL DE ARCHIVO */
  (6)  GRTOBJAUT OBJ(&ODLBNM/&ODOBNM) OBJTYPE(&ODOBTP) +
       USER(&USER) AUT(*CHANGE)
       GOTO READ                    /*VOLVER AL SIGUIENTE REGISTRO        */
       ENDPGM
```

(1) El archivo declarado, QADSPOBJ en QSYS, es el archivo suministrado por IBM que utiliza el mandato DSPOBJD. Este archivo es el archivo primario al cual hace referencia el mandato en el momento de crear el archivo de salida. El compilador CL le hace referencia para determinar el formato de los registros y para declarar variables para los campos en el formato de registro.

(2) El mandato DSPOBJD crea un archivo denominado DSPOBJD en la biblioteca QTEMP. Este archivo tiene el mismo formato que el archivo QADSPOBJ.

(3) El mandato OVRDBF altera temporalmente el archivo de declaración (QADSPOBJ) con el archivo creado por el mandato DSPOBJD.

(4) El mandato RCVF lee un registro desde el archivo DSPOBJD. Los valores de los campos en el registro se copian en las variables CL correspondientes, que el mandato DCLF declaró implícitamente. Debido a que ha utilizado el mandato OVRDBF, se lee el archivo QTEMP/DSPOBJD en vez de QSYS/QADSPOBJ (no se lee el archivo QSYS/QADSPOBJ).

(5) Se supervisa el mensaje CPF0864. Ello indica que se ha alcanzado el final del archivo, por lo que el procedimiento devuelve el control al procedimiento de llamada.

(6) Se procesa el mandato GRTOBJAUT, utilizando las variables para nombre de objeto, nombre de biblioteca y tipo de objeto, que leyó el mandato RCVF.

Nota: QUSRTOOL proporciona igualmente varios mandatos (CVTxxx) que crean archivos de salida.

6.0 Capítulo 6. Temas de Programación Avanzada

Este capítulo explica más temas de programación avanzada, que incluyen:

- Las funciones especiales que pueden llamarse desde programas en lenguajes de alto nivel (incluyendo los programas CL)
- Utilización de las solicitudes y el Menú del Programador para entrar el fuente del programa

Consulte la publicación System API Reference para obtener información referente al proceso de mandatos de función avanzada.

Este capítulo incluye Interfaces de Programación de Uso General (GUPI), que proporciona IBM para su uso en programas elaborados por el cliente.

Al final del capítulo se incluyen varios programas de ejemplo.

Subtemas

6.1 Utilización del Programa QCAPCMD

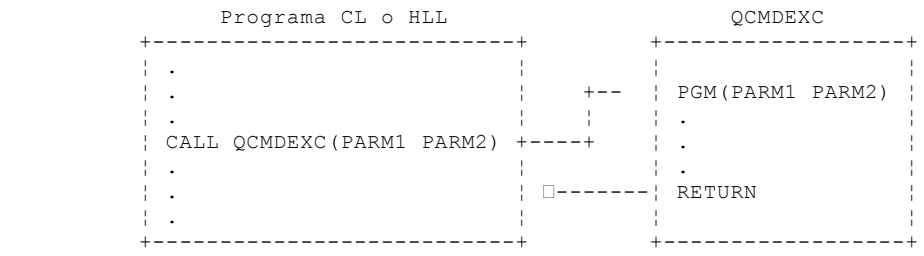
La API Procesar Mandatos (QCAPCMD) ejecuta el proceso analizador del mandato en series de mandato. Puede utilizar esta API para efectuar las siguientes acciones:

- Comprobar la sintaxis de una serie de mandato antes de ejecutarlo.
- Solicitar el mandato y recibir la serie del mandato cambiada.
- Utilizar un mandato en un lenguaje de alto nivel.
- Visualizar la ayuda para un mandato.

|Consulte el manual System API Reference, SC41-4801 para obtener información referente a la API QCAPCMD.

6.2 Utilización del Programa QCMDEXC

QCMDEXC es un programa proporcionado por IBM que ejecuta un solo mandato. Se utiliza para ejecutar un mandato desde un programa en lenguaje de alto nivel (HLL) o desde un procedimiento o programa CL en el que, en el momento de la compilación, se desconoce qué mandato se ejecutará o qué parámetros se van a utilizar. El programa QCMDEXC se llama desde su procedimiento o programa HLL o CL y el mandato que ejecuta se le pasa como un parámetro en el mandato CALL.



Después de la ejecución del mandato, se devuelve el control a su programa o procedimiento HLL o CL.

El mandato se ejecuta como si no estuviera en un programa. Por consiguiente, no pueden utilizarse variables en el mandato; el mandato no puede devolver valores a las variables CL; y los mandatos que sólo pueden utilizarse en procedimientos o programas CL no pueden ejecutarse en el programa QCMDEXC. El formato de la llamada al programa QCMDEXC es el siguiente:

```
  CALL PGM(QCMDEXC) PARM(mandato longitud-mandato)
```

El mandato que desea ejecutar se entra como una serie de caracteres en el primer parámetro. Si el mandato contiene espacios en blanco, debe escribirse entre apóstrofos. La longitud máxima de la serie de caracteres es de 6000 caracteres; los delimitadores (los apóstrofos que encierran la serie) no se cuentan nunca como parte de la serie. La longitud especificada como segundo valor en el parámetro PARM es la longitud de la serie de caracteres que se pasa como mandato. Esta longitud debe ser un valor decimal empaquetado con una longitud de 15 con 5 posiciones decimales.

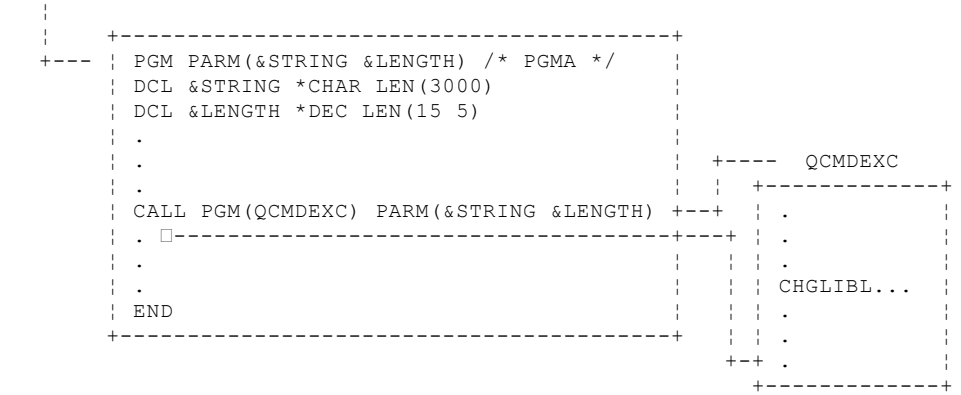
Así pues, si desea sustituir una lista de biblioteca, la llamada al programa QCMDEXC sería la siguiente:

```
  CALL PGM(QCMDEXC) PARM('CHGLIBL LIBL(QGPL NEWLIB QTEMP)' 31)
```

Esta sentencia puede codificarse en un programa HLL o CL y, cuando el programa se ejecutase, se sustituiría la lista de bibliotecas. De esta forma, sin embargo, la utilización del programa QCMDEXC no proporciona flexibilidad a la hora de ejecutarlo. Esto se consigue sustituyendo las variables por constantes en la lista de parámetros y especificando los valores para las variables en la llamada al programa HLL o CL.

Por ejemplo:

```
CALL PGM(PGMA) PARM('CHGLIBL...' 3000)
```



La longitud del mandato, que se pasa al programa QCMDEXC en el segundo parámetro, es la longitud máxima de la serie del mandato que se pasa. Si la serie del mandato se pasa como una serie entre comillas, la longitud del mandato es exactamente la longitud de la serie entrecomillada. Si la serie entrecomillada se pasa en una variable, la longitud de mandato es la longitud de la variable CL. No es necesario reducir la longitud del mandato a la longitud real de la serie del mandato en la variable, aunque es posible hacerlo.

Se puede ejecutar el mismo mandato utilizando el siguiente mandato CALL desde el procedimiento o programa CL porque los literales numéricos pasan como valores decimales empaquetados con una longitud de 15 con 5 posiciones decimales:

```
CALL QCMDEXC (&STRING 3000)
```

No todos los mandatos pueden ejecutarse utilizando el programa QCMDEXC. El mandato pasado en una llamada al programa QCMDEXC debe ser válido en el entorno actual (interactivo o por lotes) en el que se realiza la llamada. El mandato no puede ser uno de los siguientes:

- Un mandato de control de corriente de entrada (BCHJOB, ENDBCHJOB y DATA)
- Un mandato que puede utilizarse solamente en un programa CL

Para determinar si un mandato CL puede pasarse en una llamada al programa QCMDEXC, consulte el mandato en el manual CL Reference. Para averiguar cuál es el entorno en el que se puede ejecutar el mandato, consulte el diagrama de sintaxis de dicho mandato. El recuadro pequeño situado en el ángulo superior derecho del diagrama indica cuál es el entorno en el que se puede ejecutar el mandato. Por ejemplo,**TRABAJO: I** indica que el mandato se puede procesar de forma interactiva; **TRABAJO: B** indica que el mandato se puede procesar en un trabajo por lotes; Exec indica que el mandato se puede procesar con QCMDEXC.

Puede preceder el mandato CL con un signo de interrogación (?) para pedir la solicitud o utilizar la solicitud interactiva cuando llame a QCMDEXC en un trabajo interactivo.

Si se detecta un error mientras se procesa un mandato mediante el programa QCMDEXC, se envía un mensaje de escape. Se puede supervisar este mensaje de escape en el procedimiento o programa CL utilizando el mandato Supervisar Mensaje (MONMSG). Para más información sobre la supervisión de mensajes, vea el Capítulo 7 y el Capítulo 8.

Si se detecta un error de sintaxis, se envía el mensaje CPF0006. Si se detecta un error durante el proceso de un mandato, el programa QCMDEXC devuelve cualquier mensaje de escape enviado por el mandato. Los mensajes de mandatos ejecutados mediante el programa QCMDEXC se supervisan del mismo modo que los mensajes de mandatos contenidos en procedimientos o programas CL.

Consulte el manual de consulta de HLL adecuado para obtener información sobre cómo los programas HLL manejan errores en las llamadas.

Subtemas
6.2.1 Utilización del Programa QCMDEXC con Datos DBCS

6.2.1 Utilización del Programa QCMDEXC con Datos DBCS

Puede utilizar QCMDEXC para solicitar datos de entrada del juego de caracteres de doble byte (DBCS) que se entrarán con un mandato. El formato del mandato utilizado para que QCMDEXC solicite datos de doble byte es:

```
CALL QCMDEXC ('mandato' longitud-mandato IGC)
```

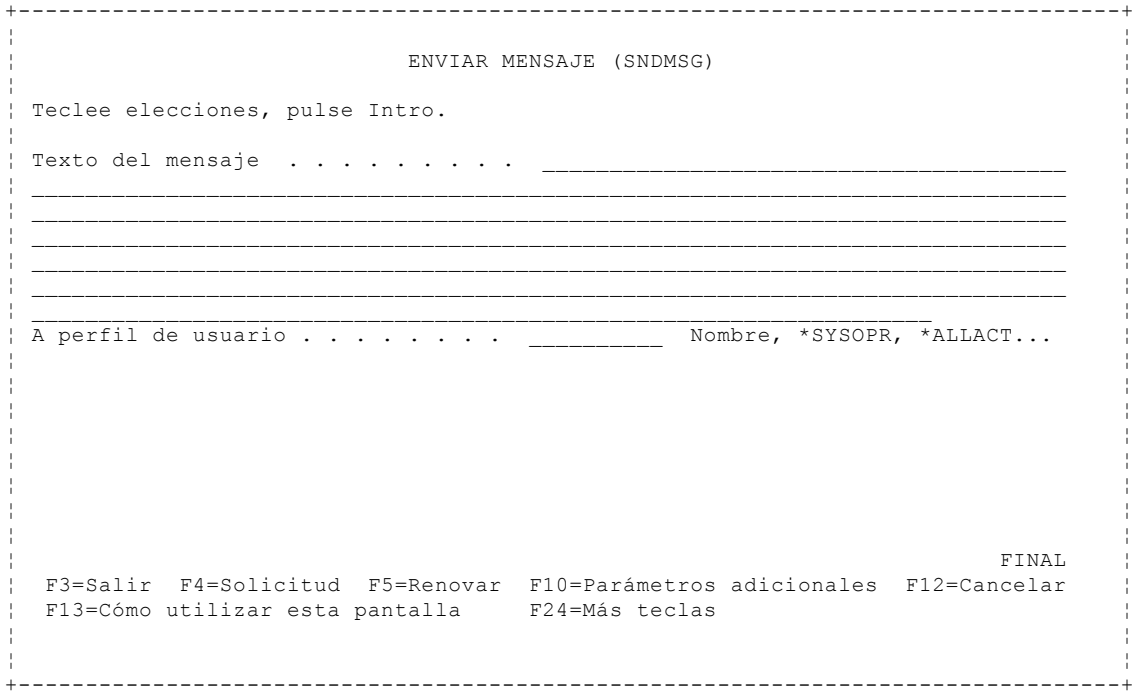
El tercer parámetro del programa QCMDEXC, IGC, indica al sistema que acepte datos de doble byte. Por ejemplo, el programa CL siguiente pide a un usuario que proporcione texto de doble byte para un mensaje. Entonces el sistema envía el mensaje siguiente:

```
PGM
   CALL QCMDEXC ('?SNDMSG' 7 IGC)
ENDPGM
```

A continuación se facilita una explicación del mensaje del sistema:

- El carácter **?** indica al sistema que presente la solicitud del mandato para Enviar Mensaje (SNDMSG).
- El valor **7** es la longitud del mandato SNDMSG más el signo de interrogación.
- El valor **IGC** permite solicitar datos de doble byte.

La pantalla siguiente se visualiza después de ejecutar el programa QCMDEXC. Puede utilizar la conversión de doble byte en esta pantalla:



6.3 Utilización del Programa QCMDCHK

QCMDCHK es un programa proporcionado por IBM que lleva a cabo la comprobación de sintaxis de un solo mandato y que, de forma opcional, solicita el mandato. El mandato no se ejecuta. Si se pide una solicitud, la serie del mandato se devuelve al procedimiento o programa de llamada con los valores actualizados tal como se entraron mediante la solicitud. El programa QCMDCHK puede llamarse desde un procedimiento o programa CL o desde un procedimiento o programa HLL.

Los usos más corrientes de QCMDCHK son los siguientes:

- Solicitar al usuario un mandato y almacenarlo para procesarlo posteriormente.
- Determinar las opciones que el usuario especificó.
- Anotar cronológicamente el mandato procesado. En primer lugar, el mandato procesado se solicita con QCMDCHK, se ejecuta con QCMDEXC y, por último, se anota cronológicamente.

El formato de la llamada a QCMDCHK es:

```
CALL PGM(QCMDCHK) PARM(mandato  longitud-mandato)
```

El primer parámetro pasado a QCMDCHK es una serie de caracteres que contiene el mandato a comprobar o solicitar. Si el primer parámetro es una variable y se pide la solicitud, el mandato que entre el usuario de la estación de trabajo se coloca en la variable.

El segundo parámetro es la longitud máxima de la serie del mandato que se pasa. Si la serie del mandato se pasa como una serie entre comillas, la longitud del mandato es exactamente la longitud de la serie entrecomillada. Si la serie entrecomillada se pasa en una variable, la longitud de mandato es la longitud de la variable CL. El segundo parámetro debe ser un valor decimal empaquetado de una longitud de 15 con 5 posiciones decimales.

El programa QCMDCHK realiza comprobaciones de sintaxis en la serie del mandato que se pasa a dicho programa. Verifica que todos los parámetros necesarios estén codificados y que todos los parámetros tengan valores permitidos. No comprueba el entorno del proceso. Es decir que, un mandato puede comprobarse si se permite solamente en lotes, sólo interactivamente o solamente en un programa CL por lotes o interactivo. QCMDCHK no permite la comprobación de las sentencias de definición de un mandato.

Si se detecta un error de sintaxis en el mandato, se envía un mensaje CPF006. Puede controlar si aparece este mensaje para determinar si se ha producido un error en el mandato. El mensaje CPF006 va precedido por uno o más mensajes de diagnóstico que identifican el error. En el ejemplo siguiente, se pasa el control a la etiqueta ERROR en el programa, porque el valor 123 no es válido para el parámetro PGM del mandato CRTCLPGM.

```
CALL QCMDCHK ('CRTCLPGM PGM(QGPL/123)' 22)
MONMSG CPF0006 EXEC(GOTO ERROR)
```

Puede pedir una solicitud para el mandato colocando un signo de interrogación delante del nombre del mandato o colocando caracteres de solicitud selectiva delante de uno o más nombres de palabras clave en la serie del mandato.

Si no se detectan errores durante la comprobación y solicitud del mandato, la serie del mandato actualizada se coloca en la variable especificada para el primer parámetro. Los caracteres de petición de solicitud se eliminan de la serie del mandato. Esto se muestra en el ejemplo siguiente:

```
DCL &CMD *CHAR 2000
 .
 .
CHGVAR &CMD '?CRTCLPGM'
CALL QCMDCHK (&CMD 2000)
```

Después de ejecutarse la llamada al programa QCMDCHK, la variable &CMD contiene la serie del mandato con todos los valores entrados mediante la solicitud. Podría ser algo así:

```
CRTCLPGM PGM(PGMA) SRCFILE(TESTLIB/SOURCE) USRPRF(*OWNER)
```

Observe que el signo de interrogación que precede al nombre del mandato se elimina.

Cuando se pide una solicitud mediante el programa QCMDCHK, la serie del mandato debe pasarse en una variable CL. De lo contrario, la serie del mandato actualizada no se devuelve a su procedimiento o programa. También debe asegurarse de que la variable para la serie del mandato sea de un tamaño suficiente para contener la serie del mandato actualizada que se

devuelve desde la solicitud. Si no es así, se envía el mensaje CPF0005 y no se cambia la variable que contiene la serie del mandato. Sin la solicitud selectiva, la solicitud devuelve solamente las entradas que ha tecleado el usuario.

La longitud de la variable está determinada por el valor del segundo parámetro y no por la longitud real de la variable. En el ejemplo siguiente, se envía el mensaje de escape CPF0005 porque la longitud especificada no es suficiente para contener el mandato actualizado, aunque la variable se declarara con una longitud adecuada.

```
  DCL &CMD *CHAR 2000
   .
   .
  CHGVAR &CMD '?CRTCLPGM'
  CALL QCMDCHK (&CMD 9)
```

Si pulsa F3 ó F12 para dejar de utilizar las solicitudes mientras se procesa QCMDCHK, el mensaje CPF6801 se envía al procedimiento o programa que llamó QCMDCHK, y la variable que contiene la serie del mandato no se modifica.

Si se especifica PASSATR(*YES) en la sentencia de definición de mandato PARM, ELEM o QUAL, y se cambia el valor por omisión utilizando el mandato CHGCMDDFT, el valor por omisión se resalta como si fuera un valor especificado por el usuario, y no un valor por omisión. Si el valor por omisión de una sentencia de definición del mandato PARM, ELEM o QUAL modificada se cambia por el valor por omisión original, el valor por omisión ya no estará resaltado.

6.4 Utilización de Subarchivos de Mensajes en un Programa o Procedimiento CL

En procedimientos o programas CL, los subarchivos de mensajes son el único tipo de subarchivos soportados. Para utilizar el soporte de subarchivos de mensajes, ejecute un mandato SNDF o SNDRCVF utilizando el registro de control de subarchivos de mensajes. En las DDS, facilite datos a SFLPGMQ y SFLINZ siempre estará activa.

Cuando utiliza subarchivos de mensajes en procedimientos o programas CL, debe especificar un procedimiento o programa. No se puede especificar un asterisco para la palabra clave SFLPGMQ en las DDS. Cuando especifica un nombre de procedimiento o programa OPM, todos los mensajes enviados a esa cola de mensajes del procedimiento o programa se extraen de la cola de mensajes de invocación y se ubican en el subarchivo de mensajes. Todos los mensajes asociados con la petición actual se toman de la cola de mensajes CALL y se colocan en el subarchivo de mensajes.

Los subarchivos de mensajes permiten que un procedimiento o programa de control visualice uno o más mensajes de error.

6.5 Permitir al Usuario Cambiar Mandatos CL en la Ejecución

Con la mayoría de los procedimientos o programas CL, el usuario de la estación de trabajo proporciona entrada al procedimiento o programa especificando valores de parámetros pasados al procedimiento o programa, o tecleando en campos con posibilidad de entrada en una pantalla de solicitud.

También puede solicitar entrada al usuario de la estación de trabajo para un procedimiento o programa CL, de las siguientes maneras:

- Si se entra un ? antes del mandato CL en el fuente del programa o procedimiento CL, el sistema visualiza una solicitud para el mandato CL. Los valores de parámetro ya especificados en su procedimiento o programa se rellenan, y el usuario de la estación de trabajo no puede cambiarlos. Véase la sección "Utilización de Solicitudes en un Procedimiento o Programa CL" en el tema 6.5.1 más adelante en este apartado.
- Si llama al programa QCMDEXC y pide la solicitud selectiva, el sistema visualiza una solicitud para un mandato CL, pero no es necesario que especifique en el fuente del programa CL qué mandato CL se va a utilizar en el proceso. Para obtener más información sobre el programa QCMDEXC, véase el apartado "Utilización del Programa QCMDEXC" en el tema 6.2.

Subtemas

6.5.1 Utilización de Solicitudes en un Procedimiento o Programa CL

Puede pedir el uso de solicitudes en el proceso interactivo de un procedimiento o programa CL. Por ejemplo, puede compilarse y ejecutarse el siguiente procedimiento:

```
PGM
.
.
.
?DSPLIB
.
.
.
ENDPGM
```

En este caso, la solicitud para el mandato Visualizar Biblioteca (DSPLIB) aparece en la pantalla durante el proceso del programa. El proceso del mandato DSPLIB espera hasta que el usuario ha entrado los valores de los parámetros requeridos y ha pulsado la tecla Intro.

No puede cambiarse ningún valor especificado en el procedimiento fuente por parte del operador (o usuario). Por ejemplo:

```
PGM
.
.
.
?SNDMSG TOMSGQ(WS01 WS02)
.
.
.
ENDPGM
```

Cuando se llama al procedimiento y aparece la solicitud para el mandato Enviar Mensaje (SNDMSG), el operador (o usuario) puede entrar valores en los parámetros MSG, MSGTYPE y RPYMSGQ, pero no puede alterar los valores del parámetro TOMSGQ. Por ejemplo, el operados (o usuario) no puede añadir WS03 ni suprimir WS02. Consulte el apartado "QCMDEXC con Solicitudes en Procedimientos o Programas CL" en el tema 6.5.3 para ver una excepción a esta restricción. En el momento del proceso se aplican las siguientes restricciones al uso de solicitudes en un procedimiento CL:

- Cuando se utilizan solicitudes en un procedimiento o programa CL, no puede entrar un nombre de variable ni una expresión para un valor de parámetro en la solicitud.

- No se puede pedir la utilización de solicitudes en un mandato que se encuentra en un mandato IF, ELSE o MONMSG:

  Correcto

  ```
  IF (&A=5) THEN(DO)
              ?SNDMSG
              ENDDO
  ```

  Incorrecto

  ```
  IF (&A=5) THEN(?SNDMSG)
  ```

- *No se pueden utilizar* solicitudes en los mandatos siguientes:

  ```
  CALLPRC
  CALL        ENDDO         PGM
  CHGVAR      ENDPGM        RCVF
  DCL         ENDRCV        RETURN
  DCLF        IF            SNDF
  DO          GOTO          SNDRCVF
  ELSE        MONMSG        WAIT
  ```

- No pueden utilizarse solicitudes en trabajos por lotes.

Cuando entra una petición de solicitud (?) en un mandato de un miembro de archivo fuente CL, puede que reciba un mensaje de diagnóstico en el mandato y siga teniendo una compilación satisfactoria. En este caso, debe examinar los mensajes cuidadosamente para ver qué errores pueden corregirse mediante valores entrados en la pantalla de solicitud cuando se ejecuta el procedimiento o programa.

Puede solicitar todos los mandatos sobre los que dispone de autorización en cualquier modalidad mientras estén en un entorno interactivo, a excepción de los mandatos listados anteriormente, los cuales no pueden realizar una solicitud durante el proceso de un procedimiento o programa CL. Ello le permite utilizar una solicitud de cualquier mandato mientras esté en una estación de trabajo, y reduce la necesidad de consultar los

manuales que describen los diversos mandatos y sus parámetros.

Si pulsa F3 o F12 para cancelar el mandato con solicitud durante su ejecución, se enviará un mensaje de escape (CPF6801) al procedimiento o programa CL. Puede supervisar este mensaje utilizando el mandato MONMSG en el procedimiento o programa CL.

Cuando realiza una solicitud para un mandato, su procedimiento o programa no recibe la serie de mandato que entró. Para ello, efectúe la solicitud
|utilizando QCMDCHK y después ejecute el mandato mediante QCMDEXC. También
|puede utilizar QCAPCMD para solicitar y ejecutar el mandato.

6.5.2 Solicitud Selectiva de Mandatos CL

Puede pedir solicitudes para parámetros seleccionados en un mandato. Esto resulta especialmente útil cuando se utilizan algunos de los mandatos de mayor longitud y no desea que se le soliciten determinados parámetros.

La solicitud selectiva se puede utilizar durante la solicitud interactiva, o se puede entrar como fuente (en SEU) para usarla dentro de un procedimiento o programa CL. Puede introducir el fuente para la solicitud selectiva con SEU pero no puede utilizar la solicitud selectiva al entrar mandatos en SEU.

La solicitud selectiva se puede utilizar para:

- Seleccionar los parámetros para los que es necesaria la solicitud.
- Determinar qué parámetros están protegidos.
- Omitir parámetros de la solicitud.

A la solicitud selectiva se aplican las siguientes restricciones:

- El nombre o la etiqueta del mandato debe irá precedida por un **?** (signo de interrogación):

  - Cuando uno o más de las opciones de solicitud selectiva sea **?-** (interrogante, menos).

  - Para evitar obtener un mensaje CPF6805 (mensaje que indica la existencia de un problema de diagnóstico en el mandato aunque se lleve a cabo la compilación).

- Los parámetros pueden especificarse por la posición pero no pueden ir precedidos por caracteres de solicitud selectiva.

- Un parámetro debe adoptar la forma de palabra clave para ser solicitado selectivamente.

- No pueden entrarse espacios en blanco entre los caracteres de solicitud selectiva y la palabra clave.

- La solicitud selectiva se aplica solamente a nivel de parámetro; es decir, no puede especificar valores de una palabra clave determinada en una lista de valores.

- **?-** no está permitido en programas de alteración temporal de solicitud.

- Si un parámetro es necesario, debe utilizarse la solicitud selectiva **??**.

  Puede saberse que un parámetro es necesario porque la posición de entrada se resalta cuando se solicita el mandato.

Los valores especificados por el usuario se marcan con un símbolo especial (**>**) delante de los valores tanto en la solicitud selectiva como en la normal. Si un valor especificado por el usuario en la solicitud del parámetro no va precedido de este símbolo, el mandato por omisión se pasa al programa de proceso de mandatos.

Si se especifica PASSATR(*YES) en las sentencias de definición de mandato PARM, ELEM o QUAL, y el valor por omisión se modifica mediante el mandato CHGCDMDFT, el valor por omisión aparece como un valor especificado por el usuario (con el símbolo >) y no como un valor por omisión. Si el valor por omisión de una sentencia de definición del mandato PARM, ELEM o QUAL modificada se cambia por el valor por omisión original, se suprime el símbolo >.

Puede pulsar F5 mientras se utiliza la solicitud selectiva para visualizar de nuevo los valores mostrados inicialmente en la pantalla.

Si utiliza una variable CL para especificar un valor para un parámetro que va a visualizarse mediante la solicitud selectivo, se puede modificar el valor en la solicitud y se utiliza el valor modificado cuando se ejecuta el mandato. El valor de la variable del procedimiento o programa no se modifica. Si un procedimiento CL contiene lo siguiente:

```
  OVRDBF ?*FILE(FILEA) ??TOFILE(&FILENAME) ??MBR(MBR1)
```

Los tres parámetros, FILE, TOFILE, y MBR aparecerán en la pantalla de solicitud. No le será posible modificar el valor especificado para el parámetro FILE, pero sí los valores para los parámetros TOFILE y MBR. Se supone que la variable CL &FILENAME tiene un valor de FILE1, y que lo cambia por FILE2. Cuando se ejecuta el mandato, se utiliza el valor de FILE2, pero el valor de &FILENAME no se cambia en el procedimiento. Las

tablas siguientes muestran los diversos caracteres de solicitud selectiva así como la acción resultante.

Si entra	Valor visualizado	Protegido	Valor pasado a CPP si no se especifica nada	Marcado con símbolo >
??KEYWORD()	Valor por omisión	No	Valor por omisión	No
??KEYWORD(VALOR)	Valor	No	Valor	Sí
?*KEYWORD()	Valor por omisión	Sí	Valor por omisión	No
?*KEYWORD(VALOR)	Valor	Sí	Valor	Sí
?<KEYWORD()	Valor por omisión	No	Valor por omisión	No
?<KEYWORD(VALOR)	Valor	No	Valor por omisión	No
?/KEYWORD()	Valor por omisión	Sí	Valor por omisión	No
?/KEYWORD(VALOR)	Valor	Sí	Valor por omisión	No
?-KEYWORD()	Ninguno	N/A	Valor por omisión	N/A
?-KEYWORD(VALOR)	Ninguno	N/A	Valor	N/A
?&KEYWORD()	Valor por omisión	No	Valor por omisión	No
?&KEYWORD(VALUE)	Valor	No	Valor por omisión	No
?%KEYWORD()	Valor por omisión	Sí	Valor por omisión	No
?%KEYWORD(VALOR)	Valor	Sí	Valor por omisión	No

Si entra	Valor de pantalla al pulsar F5 o borrar	Descripción
??KEYWORD()	Valor por omisión	Solicitud de palabra clave normal con mandato por omisión.
??KEYWORD(VALOR)	Valor	Solicitud de palabra clave normal con valor por omisión especificado por el programa.
?*KEYWORD()	Valor por omisión	Mostrar solicitud protegida (como información) donde el valor por omisión del mandato sea el único valor utilizado.
?*KEYWORD(VALOR)	Valor	Mostrar solicitud protegida (como información) donde el valor especificado por el programa es el único valor usado. Por ejemplo, cuando un valor debe mostrarse como información pero no cambiarse.
?<KEYWORD()	Valor por omisión	Solicitud de palabra clave normal con mandato por omisión.
?<KEYWORD(VALOR)	Valor	Solicitud de palabra clave normal con valor por omisión especificado por el programa.
?/KEYWORD()	Valor por omisión	Reservado para utilización de IBM.

?/KEYWORD(VALOR)	Valor	Reservado para utilización de IBM.
?&KEYWORD()	Valor por omisión	Solicitud de palabra clave normal con mandato por omisión.
?&KEYWORD(VALUE)	Valor	Solicitud de palabra clave normal con valor por omisión especificado por el programa.
?%KEYWORD()	Valor por omisión	Mostrar solicitud protegida (como información) donde el valor por omisión del mandato sea el único valor utilizado.
?%KEYWORD(VALOR)	Valor	Mostrar solicitud protegida (como información) donde el valor especificado por el programa es el único valor usado. Por ejemplo, cuando un valor debe mostrarse como información pero no cambiarse.

La solicitud selectiva puede utilizarse con el programa QCMDEXC o QCMDCHK. El formato de la llamada es:

```
CALL PGM(QCMDEXC o QCMDCHK) PARM(mandato longitud-mandato)
```

A continuación se ofrece una breve descripción de los caracteres de solicitud selectiva:

Carácter de solicitud selectiva	**Descripción**
??	El parámetro se visualiza y es susceptible de entrada.
?*	El parámetro se visualiza pero no es susceptible de entrada. Cualquier valor especificado por el usuario se pasa al programa de proceso de mandatos.
?<	El parámetro se visualiza y es susceptible de entrada, pero el valor por omisión del mandato se envía al CPP a menos que se cambie el valor visualizado en el parámetro.
?/	Reservado para utilización de IBM.
?-	El parámetro no se visualiza. El valor especificado (o valor por omisión) se pasa al CPP. No autorizado en programas de alteración temporal de solicitud.
?&	El parámetro no se visualiza hasta que se pulsa F9=Todos los parámetros. Una vez visualizado, es susceptible de entrada. El valor por omisión del mandato se envía al CPP a menos que se cambie el valor visualizado en el parámetro.
?%	El parámetro no se visualiza hasta que se pulsa F9=Todos los parámetros. Una vez visualizado, no es susceptible de entrada. El valor por omisión del mandato se envía al CPP.

Para obtener información más detallada sobre QCMDEXC o QCMDCHK, consulte los apartados "Utilización del Programa QCMDEXC" en el tema 6.2 y "Utilización del Programa QCMDCHK" en el tema 6.3.

6.5.3 QCMDEXC con Solicitudes en Procedimientos o Programas CL

El programa QCMDEXC se puede utilizar para realizar solicitudes. Esta utilización de QCMDEXC con solicitudes en procedimientos o programas CL, le permite alterar todos los valores del mandato a excepción del nombre de mandato. Resulta más flexible que el uso de las solicitudes directamente, en que sólo puede entrar valores no especificados en el fuente (véase el apartado anterior). Si se utiliza una solicitud directamente en un mandato, como se indica a continuación:

```
  ?OVRDBF FILE(FILEX)
```

puede especificar un valor para cualquier parámetro, excepto FILE. Sin embargo, si se llama al mandato durante el proceso de un programa utilizando el programa QCMDEXC, tal como:

```
  CALL QCMDEXC PARM('?OVRDBF FILE(FILEX)' 19)
```

puede especificar un valor para cualquier parámetro, incluyendo FILE. En este ejemplo, FILEX es el valor por omisión.

También se pueden conseguir solicitudes con valores especificado modificables utilizando la solicitud selectiva, tal como se describe anteriormente en este capítulo. Sin embargo, cada palabra clave que se desee utilizar debe ser seleccionada explícitamente. Se llamará a la solicitud directamente con un mandato como:

```
  OVRDBF ??FILE(FILEX) ??TOFILE(*N) ??MBR(*N)
```

6.6 Utilización del Menú del Programador

El menú del programador puede llamarse directamente llamado al programa QPGMMENU o utilizando el mandato Arrancar Menú del Programador (STRPGMMNU). Puede utilizar este mandato para especificar por adelantando los valores por omisión que utilizará con el menú del programador. Además, el mandato STRPGMMNU también da soporte a otras opciones que se pueden utilizar para adaptar dicho menú.

Para ver una descripción del mandato STRPGMMNU y sus parámetros, consulte el manual CL Reference.

Subtemas
6.6.1 Usos del Mandato Arrancar Menú del Programador (STRPGMMNU)

6.6.1 Usos del Mandato Arrancar Menú del Programador (STRPGMMNU)

El mandato Arrancar Menú del Programador puede utilizarse para:

- Realizar la misma función que una llamada a QPGMMENU
- Rellenar los campos de entrada estándares

  Cuatro de los parámetros del mandato le permiten rellenar los campos de entrada estándares al final del menú. Estos parámetros son los siguientes:
  - Archivo fuente
  - Biblioteca fuente
  - Biblioteca objeto
  - Descripción del trabajo

Este mandato puede utilizarse con uno o más de los parámetros que controlan los valores iniciales del menú. Puede diseñarse esto como parte de un programa inicial para el inicio de sesión o para aquellas situaciones en las que un usuario llama a una función específica escrita por el usuario. El ejemplo siguiente muestra un programa de este tipo, con una función separada para cada área de aplicación que requiera valores iniciales distintos.

```
        PGM
        CHGLIBL   LIBL(PGMR1 QGPL QTEMP)
  LOOP:
        STRPGMMNU SRCLIB(PGMR1) OBJLIB(PGMR1) JOBD(PGMR1)
        MONMSG    MSGID(CPF2320) EXEC(GOTO END) /* F3 o F12 para salir del menú */
        GOTO LOOP
  END:  ENDPGM
```

- Controlar las opciones del menú del programador

  Los demás parámetros le ayudan a controlar el menú y sus funciones. Por ejemplo, puede especificar ALWUSRCHG(*NO) para evitar que un usuario cambie los valores que aparecen en el menú. Este parámetro no debe considerarse como una característica de seguridad, ya que un usuario que utilice el menú puede llamar al mandato STRPGMMNU y cambiar los valores en una llamada distinta (el usuario también puede arrancar funciones utilizando F10 para llamar la pantalla de entrada de mandatos). Si el menú se visualiza mediante el mandato STRPGMMNU, se puede evitar que el usuario (por autorización) llame al programa QPGMMENU directamente, pero no se puede evitar que el usuario solicite otra llamada al mandato STRPGMMNU.

- Adaptar la opción de creación de menú

  Los parámetros EXITPGM y DLTOPT le permiten proporcionar su propio soporte para la opción de creación del menú (opción 3). Puede llamarse a un programa de usuario cuando se solicita la opción 3. Para un amplio tratamiento de los parámetros y la lista de parámetros pasada al programa de usuario, consulte el manual CL Reference. A continuación se describen algunas utilizaciones típicas del parámetro EXITPGM.

Subtemas
6.6.1.1 El Parámetro EXITPGM

6.6.1.1 El Parámetro EXITPGM

El parámetro EXITPGM se puede utilizar con los siguientes fines:

- Para cambiar los valores por omisión utilizados en los mandatos de creación sometidos por la opción 3.

  Por ejemplo, si no se utiliza F4 (Solicitud), el parámetro EXITPGM puede cambiar uno o varios mandatos de creación para especificar los requisitos por omisión que desee el usuario. Si se utiliza F4, el parámetro EXITPGM puede someter el mandato como si lo hubiera entrado el programador (sin cambiar ningún parámetro).

- Para cambiar parámetros independientemente de la utilización de F4 por el programador.

  Ello requiere examinar el valor del parámetro &RQSDTA512 (que se ha pasado al programa de salida), para ver si ya se ha utilizado y sustituido el valor.

- Para cambiar otros parámetros en el mandato SBMJOB.

  Por ejemplo, el parámetro de usuario del mandato SBMJOB podría modificarse para especificar el valor de la descripción del trabajo, en vez del valor de *CURRENT. También es posible recuperar los valores de uno o más atributos del trabajo mediante el mandato RTVJOBA, entrando los atributos como valores específicos.

- Para reforzar convenios de programación local.

  Por ejemplo, si tiene una denominación estándar que requiere que el nombre de todos los archivos físicos tenga 7 caracteres y termine con P, el programa de salida puede rechazar cualquier intento de utilizar el mandato CRTPF con un nombre que no siga este estándar.

- Para facilitar la sustitución de objetos existentes, véase el miembro SBMPARMS del archivo QATTINFO en la biblioteca QUSRTOOL para obtener documentación y fuente de ejemplo para esta función y para el programa de salida STRPGMMNU.

6.7 Programación de Aplicaciones para Datos DBCS

Deben tenerse en cuenta consideraciones especiales al diseñar programas de aplicación para procesar datos de doble byte o convertir programas de aplicación alfanuméricos a programas de doble byte.

Subtemas
6.7.1 Diseño de Programas de Aplicación DBCS
6.7.2 Conversión de Programas Alfanuméricos para Procesar Datos DBCS

6.7.1 Diseño de Programas de Aplicación DBCS

Diseñe los programas de aplicación para procesar datos de doble byte de la misma manera que los programas de aplicación para procesar datos alfanuméricos, teniendo en cuenta las siguientes consideraciones adicionales:

- Identifique los datos de doble byte utilizados en los archivos de base de datos, si los hay.

- Diseñe los formatos de pantalla e impresoras que puedan utilizarse con datos de doble byte.

- Si es necesario, proporcione conversión de doble byte como un medio de entrar datos para aplicaciones interactivas. Utilice la palabra clave DDS para la conversión de doble byte (IGCCNU) para especificar la conversión DBCS en archivos de pantalla.

- Escriba mensajes de error de doble byte para que el programa los visualice.

- Especifique el proceso de ampliación de caracteres de forma que el sistema imprima y visualice todos los datos de doble byte.

- Determine qué caracteres de doble byte, si los hay, deben definirse. El manual *ADTS/400: Character Generator Utility*, SC09-1769, describe el modo de definir caracteres de doble byte para los países que soportan DBCS.

6.7.2 Conversión de Programas Alfanuméricos para Procesar Datos DBCS

Si un programa de aplicación alfanumérico utiliza archivos de pantalla descritos externamente, puede cambiar dicho programa de aplicación por un programa de aplicación de doble byte, cambiando solamente los archivos. Para convertir un programa de aplicación, efectúe lo siguiente:

1. Cree una copia duplicada de las sentencias fuente para el archivo alfanumérico que desea cambiar.

2. Cambie las constantes y literales alfanuméricos a constantes y literales de doble byte.

3. Cambie los campos del archivo a uno de los siguientes tipos de datos para entrar datos DBCS:

   - Tipo de datos DBCS abierto (O)
   - Tipo de datos sólo DBCS (J)
   - Datos DBCS cualquiera (E)

   No hay que cambiar la longitud de los campos.

4. Almacene el archivo de pantalla convertido en una biblioteca separada. Dé al archivo el mismo nombre que su versión alfanumérica.

5. Para utilizar el archivo convertido en un trabajo, modifique la lista de bibliotecas utilizando el mandato Cambiar Lista de Bibliotecas (CHGLIBL), para el trabajo en el que se utiliza el archivo. La biblioteca donde se almacena el archivo de pantalla de doble byte se comprobará antes que la biblioteca donde se almacena la versión alfanumérica del archivo.

6.8 Utilización de Datos DBCS en un Programa CL

El siguiente programa muestra la utilización de distintos desplazamientos de teclado en un programa CL. Nótese que los datos de doble byte se utilizan solamente en este programa como valores de texto; los mandatos en sí constan de caracteres alfanuméricos.

Cuando se ejecuta, este programa le muestra cómo se utilizan los distintos desplazamientos de teclado para los archivos de pantalla de las DDS. Consulte el manual DDS Reference para obtener información referente a los desplazamientos de teclado de doble byte.

IMAGEN 7

6.9 Ejemplos de Programas CL

Los programas de ejemplo siguientes demuestran la flexibilidad, simplicidad y versatilidad de los programas CL. Dichos programas se describen por su función y por el usuario que probablemente los utilizará.

Subtemas

6.9.1 Programa Inicial para Puesta a Punto (Programador)

```
  PGM
  CHGLIBL LIBL(TESTLIB QGPL QTEMP)
  CHGJOB OUTQ(WSPRTR)
  TFRCTL QPGMMENU
  ENDPGM
```

La biblioteca de prueba se coloca al principio de la lista de bibliotecas, se selecciona una cola de salida para una impresora adecuada y se visualiza el menú del programador.

6.9.2 Traslado de un Objeto desde una Biblioteca de Prueba a una Biblioteca de Producción (Programador)

```
  PGM PARM(&OBJ &OBJTYPE &OPER)
  DCL &OBJ *CHAR LEN(10)
  DCL &OBJTYPE *CHAR LEN(7)
  DCL &OPER *CHAR LEN(1)  /* R=Reemplazar M=Mover */
  IF ((&OPER *NE 'M') *AND (&OPER *NE 'R')) THEN(DO)
          SNDPGMMSG MSG('Código operación debe ser "R" o "M" ')
          RETURN
          ENDDO
  IF ((&OBJTYPE *NE *PGM) *AND (&OBJTYPE *NE *FILE) *AND (&OBJTYPE +
      *NE *DTAARA)) THEN(DO)
          SNDPGMMSG MSG('Objeto' *BCAT &OBJ *BCAT ' debe ser *PGM, +
                    *FILE o *DTAARA')
          RETURN
          ENDDO
  CHKOBJ BLDLIB/&OBJ OBJTYPE(&OBJTYPE)
  MONMSG MSGID(CPF9801) EXEC(DO)
          SNDPGMMSG MSG('El objeto o tipo de objeto no existe +
                    en BLDLIB')
          RETURN
          ENDDO
  IF (&OPER *EQ 'M') THEN(DO)
          MOVOBJ BLDLIB/&OBJ OBJTYPE(&OBJTYPE) TOLIB(PRODLIB)
          MONMSG MSGID(CPF3208) EXEC(DO)
                          SNDPGMMSG MSG('El objeto' *BCAT &OBJ *BCAT ' +
                                    ya existe en PRODLIB')
                          RETURN
                          ENDDO
          CHKOBJ PRODLIB/&OBJ OBJTYPE(&OBJTYPE)
          MONMSG MSGID(CPF9801) EXEC(DO)
          SNDPGMMSG MSG('El objeto o tipo de objeto no existe +
                    en PRODLIB')
                    RETURN
                    ENDDO
          ENDDO
  RETURN
  ENDPGM
```

El nombre de objeto, el tipo de objeto y el código de operación se pasa de otro programa o procedimiento. Se realizan comprobaciones para ver si el código de operación y el tipo de objeto son correctos y si el objeto existe en la biblioteca de prueba. El objeto se mueve a menos que ya exista en la biblioteca de producción. Después se confirma el traslado. Pueden añadirse más mandatos para otorgar autorización adicional sobre el objeto o para manejar excepciones y tipos de objeto adicionales.

6.9.3 Salvar Objetos Específicos en una Aplicación (Operador del Sistema)

Subtemas
6.9.3.1 Ejemplo

6.9.3.1 Ejemplo

```
  PGM
| SAVOBJ OBJ(FILE1 FILE2) LIB(LIBA) OBJTYPE(*FILE) DEV(TAP01) +
    CLEAR(*YES)
| SAVOBJ OBJ(DTAARA1) LIB(LIBA) OBJTYPE(*DTAARA) DEV(TAP01)
  SNDPGMMSG MSG('Copia de seguridad diaria de LIBA finalizada') +
    MSGTYPE(*COMP)
  ENDPGM
```

Este programa garantiza una entrada coherente de mandatos para unos procedimientos que se repiten con regularidad.

Pueden añadirse mandatos Salvar Objeto (SAVOBJ) adicionales. Sin embargo, este programa deja que el operador seleccione el disquete o cinta correctos para cada copia periódica de seguridad de cada aplicación. Esto puede controlarse asignando nombres exclusivos a cada conjunto de disquetes o cintas para cada operación de salvar. Por ejemplo, si desea salvar sus archivos de nóminas cada semana por separado durante cuatro semanas, puede dar nombre a cada disquete o cinta de forma distinta y escribir el programa para comparar el nombre del disquete o cinta con al nombre correcto para aquella semana.

6.9.4 Recuperación de una Finalización Anómala (Operador del Sistema)

```
PGM
DCL &SWITCH *CHAR LEN(1)
RTVSYSVAL SYSVAL(QABNORMSW) RTNVAR(&SWITCH)
IF (&SWITCH *EQ '1') THEN(DO)  /*LLAMA A PROGRAMAS DE RECUPERACION*/
        SNDPGMMSG MSG('Programas recuperación en proceso. +
          No arranque subsistemas hasta que se le notifique') +
          MSGTYPE(*INFO) TOMSGQ(QSYSOPR)
        CALL PGMA
        CALL PGMB
        SNDPGMMSG MSG('Programas recuperación completados. +
          Arranque subsistemas') +
          MSGTYPE(*INFO) TOMSGQ(QSYSOPR)
        RETURN
        ENDDO
ENDPGM
```

6.9.5 Someter un Trabajo (Operador del Sistema)

```
  PGM  /*DAILYAC*/
  SBMJOB JOB(DAILYACCRC) JOBD(ACCRC2) +
              CMD(CALL ACCRC305 PARM(DAILY))
  SNDPGMMSG   MSG('Trabajo de cuentas por cobrar diarias DAILYACCRC +
              sometido por lotes') MSGTYPE(*COMP)
  ENDPGM
```

En lugar de escribir todos los parámetros para someter un trabajo, el operador del sistema solamente llama a DAILYAC.

6.9.6 Tiempo de Espera Excedido Mientras se Espera Entrada de un Dispositivo de Pantalla

```
             PGM
             DCLF FILE(QGPL/MENU)
  START:     SNDRCVF DEV(*FILE) RCDFMT(MENUFMT) WAIT(*NO)
             WAIT
             MONMSG MSGID(CPF0889) EXEC(SIGNOFF)
             CHGVAR VAR(&IN99) VALUE('0')
             IF COND(&IN01) THEN(GOTO CMDLBL(START))
  OPTION1:   /* OPCION 1-ENTRADA PEDIDOS */
             IF COND(&OPTION *EQ '1') THEN(DO)
             CALL PGM(ORDENT)
             GOTO CMDLBL(START)
             ENDDO
  OPTION2:   /* OPCION 2-VISUALIZACION PEDIDOS */
             IF COND(&OPTION *EQ '2') THEN(DO)
             CALL PGM(ORDDSP)
             GOTO CMDLBL(START)
             ENDDO
  OPTION3:   /* OPTION 3-CAMBIO PEDIDOS */
             IF COND(&OPTION *EQ '3') THEN(DO)
             CALL PGM(ORDCHG)
             GOTO CMDLBL(START)
             ENDDO
  OPTION4:   /* OPTION 4-IMPRESION PEDIDOS */
             IF COND(&OPTION *EQ '4') THEN(DO)
             CALL PGM(ORDPRT)
             GOTO CMDLBL(START)
             ENDDO
  OPTION9:   /* OPTION 9-FIN DE SESION */
             IF COND(&OPTION *EQ '9') THEN(SIGNOFF)
  OPTIONERR: /* OPCION SELECCIONADA NO VALIDA */
             CHGVAR VAR(&IN99) VALUE('1')
             GOTO CMDLBL(START)
             ENDPGM
```

Este programa muestra cómo escribir un programa CL utilizando un archivo de pantalla que esperará el tiempo que se especifique para que el usuario entre una opción. Si no lo hace, el usuario se desconectará del sistema.

El archivo de pantalla se creó con el mandato siguiente:

```
  CRTDSPF FILE(MENU) SRCFILE(QGPL/QDDSSRC) SRCMBR(MENU)  +
      DEV(*REQUESTER) WAITRCD(60)
```

El archivo de pantalla utilizará el dispositivo *REQUESTER. Cuando se emite un mandato WAIT, éste espera los segundos especificados en la palabra clave WAITRCD (60). A continuación figuran las DDS correspondientes al archivo de pantalla:

```
  SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ... ... 8

   0100       A                                      PRINT CA01(01)
   0200       A          R MENUFMT                   BLINK
   0300       A                                      TEXT('Menú entrada pedidos')
   0400       A                                    1 31'Menú entrada pedidos'
   0500       A                                    2  2'Seleccione una de estas opciones:    '
   0600       A                                    3  4'1.  Entrar pedido'
   0700       A                                    4  4'2.  Visualizar pedido'
   0800       A                                    5  4'3.  Cambiar pedido'
   0900       A                                    6  4'4.  Imprimir pedido'
   1000       A                                    7  4'9.  Fin de sesión'
   1100       A                                   23  2'Opción:'
   1200       A            OPTION         1   I 23 10
   1300       A  99                                  ERRMSG('Opción seleccionada no válida.')

                                * * * * * *  F I N  D E  F U E N T E  * * * * *
```

El programa realiza un SNDRCVF WAIT(*NO) para visualizar el menú y solicitar una opción al usuario. Entonces emite un mandato WAIT para aceptar una opción del usuario. Si el usuario introduce un valor de 1 a 4, se llama al programa adecuado. Si el usuario introduce un 9, se emite el mandato SIGNOFF. Si el usuario introduce una opción que no es válida, se visualiza el menú con el mensaje 'OPCIÓN SELECCIONADA NO VÁLIDA'. El usuario puede entrar otra opción válida. Si el usuario no responde en 60 segundos, se emite el mensaje CPF0889 al programa y el mandato MONMSG emite el mandato SIGNOFF.

Se puede utilizar un mandato SNDF que utilice un formato de registro que contenga la palabra clave INVITE DDS en lugar de SNDRCVF WAIT(*NO). La función realizada sería la misma.

6.9.7 Recuperación de Atributos de Programa

El mandato Visualizar Programa (DSPPGM) se puede utilizar para visualizar los atributos de un programa. Para recuperar algunos de los atributos (como el tipo de programa, el miembro fuente, texto o la fecha de creación) en variables CL, puede utilizarse el mandato Visualizar Descripción de Objeto (DSPOBJD) para crear un archivo de salida. El archivo de salida puede leerse en un procedimiento o programa CL, utilizando los mandatos Declarar Archivo (DCLF) y Recibir Archivo (RCVF).

|Para acceder a otros atributos del mandato DSPPGM (como USRPRF), puede
|utilizar la herramienta QUSRTOOL (RTVPGMATR) o la API de Recuperar
|Información de programa (QCLRPGMI).

6.10 Carga y Ejecución de una Aplicación desde Cintas o Disquetes

El mandato Almacenar y Ejecutar un Programa de Soporte (LODRUN) le permite al usuario almacenar y ejecutar una aplicación escrita por otro usuario o por un proveedor de software desde cintas o disquetes proporcionados por otro usuario.

Cuando se ejecuta el mandato LODRUN:

- Se busca el soporte magnético para el programa escrito por el usuario, el cual ha de recibir el nombre de QINSTAPP. Si se utiliza una cinta, ésta se rebobina primero.

- Si ya existe un programa QINSTAPP en la biblioteca QTEMP en el sistema del usuario, se borrará.

- El programa QINSTAPP se restaura en la biblioteca QTEMP mediante el mandato RSTOBJ.

- El control del sistema se pasa al programa QINSTAPP. Dicho programa se puede utilizar, por ejemplo, para restaurar otras aplicaciones en el sistema del usuario y ejecutarlas.

Subtemas
6.10.1 Responsabilidades del Transcriptor de Aplicaciones

6.10.1 Responsabilidades del Transcriptor de Aplicaciones

El usuario que proporciona el programa QINSTAPP es responsable de escribirlo y darle soporte. Este programa no lo suministra IBM*. Puede estar diseñado para llevar a cabo diversas tareas. Por ejemplo, puede:

- Restaurar y ejecutar otros programas o aplicaciones
- Restaurar un biblioteca
- Suprimir otro programa o aplicación
- Crear entornos específicos
- Corregir problemas de aplicaciones ya existentes

La Figura 6-1 muestra un ejemplo de un programa QINSTAPP. El transcriptor del programa lo salva en una cinta o disquete y lo carga en el sistema mediante el mandato LODRUN. El mandato LODRUN pasa el control del sistema al programa, que ejecuta las tareas indicadas en el programa.

```
  PGM         PARM(&DEV)  /* "Dispositivo" sólo en Parm                    */
  DCL         VAR(&DEV)   TYPE(*CHAR) LEN(10)
  DCL         VAR(&MODEL) TYPE(*CHAR) LEN(4)

  /* Puede comprobar un número de modelo adecuado, el nivel de release, etc */
  RTVSYSVAL   SYSVAL(QMODEL) RTNVAR(&MODEL)
  IF          (&MODEL *EQ 'xxxxx') THEN...

  /* Instalar una biblioteca para una nueva aplicación (programas, datos):  */
  RSTLIB      SAVLIB(NEWAPP) DEV(&DEV) ENDOPT(*LEAVE) +
                MBROPT(*ALL)
  /* Instalar un mandato para iniciar la nueva aplicación:                  */
              RSTOBJ OBJ(NEWAPP) SAVLIB(QGPL) DEV(&DEV) +
                MBROPT(*ALL)

  END:        ENDPGM
```

Figura 6-1. Ejemplo de una aplicación utilizando el mandato LODRUN

7.0 Capítulo 7. Definición de Mensajes

En el sistema AS/400, la comunicación entre procedimientos o programas, entre trabajos, entre usuarios, entre usuarios y procedimientos, y entre usuarios y programas, se efectúa a través de mensajes. Un mensaje puede ser predefinido o inmediato:

- Un mensaje predefinido se crea y existe fuera del programa que lo utiliza. Los mensajes predefinidos se almacenan en archivos de mensajes y tienen un número de mensaje. Un ejemplo de mensaje predefinido del sistema es:

  ```
  CPF0006  Errores producidos en el mandato.
  ```

- Un mensaje inmediato lo crea el emisor en el momento en que se envía y no se almacena en un archivo de mensajes. Un ejemplo de mensaje inmediato recibido en una estación de pantalla es:

  ```
  Desde . . . :   QSYSOPR      06/12/94   10:50:54
  El sistema se desactivará a las 11:00; por favor, finalice la sesión
  ```

El sistema incluye un amplio conjunto de mensajes predefinidos que permiten la comunicación entre programas en el sistema y entre el sistema y sus usuarios. Cada programa bajo licencia que se pide tiene un archivo de mensajes que se almacena en la misma biblioteca que el programa bajo licencia al que se aplica. Por ejemplo, los mensajes del sistema se almacenan en el archivo QCPFMSG de la biblioteca QSYS.

Cada mensaje predefinido en un archivo de mensajes se identifica de forma exclusiva por un código de 7 caracteres y se define por una descripción del propio mensaje. La descripción del mensaje contiene una información que abarca el texto del mensaje, el texto de ayuda del mismo, nivel de gravedad, valores de respuesta válidos y por omisión, y algunos otros atributos. Véase la descripción del mandato Añadir Descripción del Mensaje (ADDMSGD) en el manual CL Reference.

Todos los mensajes que se envían o reciben en el sistema se transmiten a través de una cola de mensajes. Los mensajes que se emiten como respuesta a una petición directa, como puede ser un mandato, se visualizan automáticamente en la pantalla desde la cual se efectuó la petición. Para todos los demás mensajes, el usuario, programa o procedimiento debe recibir el mensaje de la cola o visualizarlo. Hay varias colas de mensajes proporcionadas por IBM en el sistema, que se describen más adelante en este capítulo (véase el apartado "Tipos de Colas de Mensajes" en el tema 7.4).

El sistema también graba algunos de los mensajes que se emiten en las anotaciones cronológicas. Una anotación de trabajo contiene información relacionada con las peticiones entradas para un trabajo; la anotación histórica contiene información sobre trabajos, subsistemas y estado de dispositivos. Véase el apartado "Anotación de Mensajes" en el tema 8.7 para obtener más información sobre anotaciones.

Puede crear archivos y descripciones de mensajes propios. Si se crean mensajes predefinidos, puede utilizar el mismo mensaje en varios procedimientos o programas, pero definirlo sólo una vez. Los mensajes predefinidos pueden también modificarse y traducirse a un idioma que no sea el español (dependiendo del usuario que los visualiza) sin afectar a los procedimientos o programas que los utilizan. Si los mensajes se definieron en un procedimiento o programa, el módulo o programa debería recompilarse cuando modifique los mensajes.

Además de crear mensajes y archivos de mensajes propios, la función de manejo de mensajes del sistema permite:

- Crear y modificar las colas de mensajes (mandatos Crear Cola de Mensajes [CRTMSGQ], Modificar Cola de Mensajes [CHGMSGQ], y Trabajar con Colas de Mensajes [WRKMSGQ])
- Crear y cambiar archivos de mensajes (mandatos Crear Archivo de Mensajes [CRTMSGF], Cambiar Archivo de Mensajes [CHGMSGF])
- Añadir descripciones de mensajes (mandato Añadir Descripción de Mensaje [ADDMSGD])
- Cambiar las descripciones de mensajes (mandato Cambiar Descripción del Mensaje [CHGMSGD])
- Eliminar descripciones de mensajes (mandato Eliminar Descripción de Mensajes [RMVMSGD])
- Enviar mensajes inmediatos (mandatos Enviar Mensaje [SNDMSG], Enviar Mensaje de Interrupción [SNDBRKMSG], Enviar Mensaje de Programa [SNDPGMMSG] y Enviar Mensaje de Usuario [SNDUSRMSG])

- Visualizar mensajes y descripciones de mensajes (mandatos Visualizar Mensajes [DSPMSG], Visualizar Descripción de Mensajes [DSPMSGD], Trabajar con Mensajes [WRKMSG], y Trabajar con Descripciones de Mensajes [WRKMSGD])

- Utilizar un procedimiento o programa CL para:

  - Enviar un mensaje al usuario de una estación de trabajo o al operador del sistema (mandato Enviar Mensaje de Usuario [SNDUSRMSG])

  - Enviar un mensaje a una cola de mensajes (mandato Enviar Mensaje de Programa [SNDPGMMSG])

  - Recibir un mensaje de una cola de mensajes (mandato Recibir Mensaje [RCVMSG])

  - Enviar una respuesta para un mensaje a una cola de mensajes (mandato Enviar Respuesta [SNDRPY])

  - Recuperar un mensaje de un archivo de mensajes (Recuperar Mensaje [RTVMSG])

  - Eliminar un mensaje de una cola de mensajes (mandato Eliminar Mensaje [RMVMSG])

  - Supervisar los mensajes de escape, notificación y estado enviados a una cola de mensajes de llamada (mandato Supervisar Mensaje [MONMSG])

- Utilizar la lista de respuestas del sistema para especificar las respuestas a los mensajes de consulta predefinidos enviados por un trabajo (mandatos Añadir Entrada en Lista de Respuestas [ADDRPYLE], Cambiar Entrada en Lista de Respuestas [CHGRPYLE], Eliminar Entrada en Lista de Respuestas [RMVRPYLE] y Trabajar con Entrada en Lista de Respuestas [WRKRPYLE])

Cuando se envía un mensaje, se define como uno de los siguientes tipos:

- Informativo (*INFO). Mensaje que contiene información sobre la condición de una función.

- De consulta (*INQ). Mensaje que contiene información pero que también solicita una respuesta.

- De notificación (*NOTIFY). Mensaje que describe una condición para la cual un procedimiento o programa precisa una acción correctora o una respuesta del procedimiento o programa de llamada. Un Procedimiento o programa puede supervisar la llegada o notificación de mensajes desde los programas o procedimientos a los que llama.

- De respuesta (*RPY). Mensaje que es una respuesta a una mensaje de consulta o de notificación recibido.

- Copia del emisor (*COPY). Copia de un mensaje de consulta o de notificación que el emisor conserva.

- De petición (*RQS). Mensaje que solicita una función del procedimiento o programa receptor. (Por ejemplo, un mandato CL es un mensaje de petición.)

- De terminación (*COMP). Mensaje que transmite el estado de terminación de un trabajo.

- De diagnóstico (*DIAG). Mensaje sobre errores en el proceso de una función del sistema, en un programa de aplicación o en los datos de entrada.

- De estado (*STATUS). Mensaje que describe el estado del trabajo realizado por un procedimiento o programa. Un procedimiento o programa puede supervisar la llegada de mensajes de estado desde el procedimiento o programa que llama. Los mensajes de estado enviados a la cola de mensajes externa (*EXT) se muestran en la estación de pantalla y pueden utilizarse para informar al usuario de la estación de pantalla sobre una operación que está en curso.

- De escape (*ESCAPE). Mensaje que describe una condición por la que un procedimiento o programa debe finalizar de forma anómala. El procedimiento o programa puede supervisar la llegada de mensajes de escape desde el programa o procedimiento que llama o desde el sistema.
| El control no se devuelve al programa después de enviarse un mensaje

| de escape.

En este capítulo se describe:

- Cómo crear sus propios archivos de mensajes
- Cómo añadir descripciones de mensajes a un archivo de mensajes
- Tipos de colas de mensajes
- Cómo crear colas de mensajes

Subtemas

7.1 Creación de un Archivo de Mensajes

Para crear sus propios mensajes predefinidos, en primer lugar debe crear el archivo de mensajes en el que se colocarán los mensajes. Para ello, utilice el mandato Crear Archivo de Mensajes (CRTMSGF). A continuación utilice el mandato Añadir Descripción del Mensaje (ADDMSGD) para describir los mensajes y colocarlos en el archivo de mensajes.

En el mandato CRTMSGF, puede especificar el tamaño máximo en Kbytes del archivo en el parámetro SIZE. Puede utilizarse la fórmula siguiente para determinar dicho tamaño:

```
  S + (I x N)
```

siendo:

S la cantidad inicial de almacenamiento

I la cantidad de almacenamiento a añadir cada vez

N el número de veces que se añadirá almacenamiento

Los valores por omisión para S, I y N son 10, 2 y *NOMAX, respectivamente.

Por ejemplo, puede especificar S como 5, I como 1, y N como 2. Cuando el archivo alcanza la cantidad de almacenamiento inicial de 5 K, el sistema añade automáticamente 1 K al almacenamiento inicial. La cantidad añadida (1 K) se puede añadir al almacenamiento dos veces para alcanzar una cifra máxima de 7 K. Si se especifica *NOMAX como N, el tamaño máximo del archivo de mensajes es de 16 M.

Cuando especifique el tamaño máximo para un archivo de mensajes y éste se llene, ya no podrá cambiar el tamaño de ese archivo. Lo que tendrá que hacer es crear otro archivo de mensajes y volver a crear los mensajes en el archivo nuevo. El mandato Fusionar Archivo de Mensajes (MRGMSGF) puede utilizarse para copiar descripciones de mensajes de un archivo de mensajes a otro. Puesto que probablemente deseará ahorrarse este paso, al crear el archivo de mensajes es importante calcular el tamaño que éste necesita o especificar *NOMAX.

Subtemas
7.1.1 Determinación del Tamaño de un Archivo de Mensajes

7.1.1 Determinación del Tamaño de un Archivo de Mensajes

Puede determinar el tamaño de un mensaje utilizando la siguiente fórmula (los parámetros del mandato ADDMSGD aparecen entre paréntesis).

- El índice de mensajes es de 42 bytes de base más la longitud del mensaje.
- El texto del mensaje (MSG) es de 16 bytes de base más la longitud del mensaje.
- La información de ayuda en línea del mensaje (SECLVL) es, en caso de haberla, de 16 bytes de base más la longitud de la ayuda del mensaje.
- Los formatos (FMT), si los hay, son de 14 bytes más (3 x el número de FMTS).
- El tipo y la longitud (TYPE y LEN) son 48 bytes.
- El valor especial (SPCVAL) es de 2 más (64 x número de SPCVAL).
- Los valores (VALUES) son de 32 x (número de VALUES).
- El rango (RANGE) es de 64 bytes.
- La relación (REL) es igual a la longitud de la relación.
- El valor por omisión (DFT) es igual a la longitud de la respuesta por omisión.
- El programa por omisión, anotaciones de problemas y la lista de vuelco (DFTPBM, LOGPRB, DMPLST) son igual a 35 más (2 x número en DMPLST).
- ALROPT es igual a 12 bytes.

La entrada más pequeña posible en un archivo de mensajes es de 59 bytes y la de mayor tamaño es de 5764 bytes. En la tabla siguiente se describe la entrada más larga posible:

Índice de Mensajes	42 bytes
Texto del mensaje	148 bytes
Texto de ayuda del mensaje	3016 bytes
99 formatos	311 bytes
Tipo y longitud	48 bytes
20 valores especiales	1282 bytes
20 valores	640 bytes
Valor de respuesta por omisión	32 bytes
Programa por omisión y lista de vuelco	233 bytes
Opción de alerta	12 bytes

En el ejemplo siguiente, el mandato CRTMSGF crea el archivo de mensajes USRMSG:

```
  CRTMSGF   MSGF(QGPL/USRMSG) +
            TEXT('Archivo de mensajes para mensajes creados por el usuario')
```

|Si se crea un archivo de mensajes que se va a utilizar con el código de
|operación DSPLY en RPG para OS/400, el archivo de mensajes debe llamarse QUSERMSG.

7.2 Adición de Mensajes a un Archivo

El mandato Añadir Descripción del Mensaje (ADDMSGD) se utiliza para describir los mensajes predefinidos y para añadirlos al archivo de mensajes creado. En el mandato ADDMSGD se especifica el identificador del mensaje, el nombre del archivo de mensajes en el que se colocará el mensaje y la descripción de éste. En la descripción del mensaje, puede especificar lo siguiente:

- El texto del mensaje (necesario) con variables de sustitución opcionales.
- El texto de ayuda del mensaje con variables de sustitución opcionales.
- El código de gravedad
- La descripción del formato de los datos del mensaje que se utilizarán para las variables de sustitución
- Los valores de comprobación de validez para una respuesta
- El valor por omisión para una respuesta
- La acción de manejo de mensajes por omisión para mensajes de escape
- El nivel de creación
- Las opciones de alerta
- La entrada en el registro de errores
- ID de Juego de Caracteres (CCSID)

Cada uno de los elementos que pueden figurar en la descripción del mensaje se describen con más detalle en las páginas siguientes.

Los siguientes mandatos también se pueden utilizar con las descripciones de mensajes:

Cambiar Descripción del Mensaje (CHGMSGD)
Modifica la descripción de un mensaje.

Visualizar Descripciones de Mensajes (DSPMSGD)
Visualiza una descripción de mensaje (en este mandato puede especificar un rango de identificadores de mensaje).

Eliminar Descripción de Mensaje (RMVMSGD)
Elimina una descripción de mensaje de un archivo de mensajes.

Recuperar Mensaje (RTVMSG)
Recupera un mensaje de un archivo de mensajes.

Fusionar Archivo de Mensajes (MRGMSGF)
Fusiona mensajes de un archivo de mensajes en otro archivo de mensajes.

Trabajar con Descripciones de Mensaje (WRKMSGD)
Visualiza una lista de mensajes en un archivo y permite añadir, cambiar o suprimir las descripciones de mensaje.

Subtemas

7.2.1 Asignación de un Identificador de Mensaje

El identificador de mensaje que deberá especificar en el mandato ADDMSGD se utiliza para hacer referencia al mensaje y será el nombre de la descripción del mensaje. Dicho identificador debe constar de 7 caracteres:

```
pppmmnn
```

donde **ppp** es el código de aplicación o producto, **mm** es el código de grupo numérico y **nn** es el código de subtipo numérico. El número especificado como **mmnn** se puede utilizar para volver a dividir un conjunto de mensajes de aplicación o producto. Los códigos de subtipo y grupo numérico constan de números decimales de 0 a 9 y de los caracteres A hasta F.

Por ejemplo:

```
CPF1234
```

es el mensaje 1234 de CPF.

Cuando crea mensajes propios, la utilización de la letra U como el primer carácter del código de producto es una buena forma de distinguir estos mensajes de los mensajes del sistema. Por ejemplo:

```
USR3567
```

El primer carácter del código debe ser alfabético, mientras que el segundo y el tercero pueden ser alfanuméricos; el código de grupo y de subtipo deben estar formados por números decimales del 0 al 9 y por caracteres de la A a la F. Observe que, aunque a este rango se le puede considerar como un conjunto de números hexadecimales, una ordenación de los números de mensajes trata los caracteres de la A a la F como caracteres alfabéticos.

Por ejemplo, al visualizar un rango de descripciones de mensajes, CPFA000 aparece delante de CPF1000.

Debe tener cuidado al utilizar un código de subtipo numérico de 00 en el identificador de mensaje. Si lo emplea para un mensaje que se puede enviar como mensaje de escape, de notificación o de estado y que, por lo tanto, se pueden supervisar, este código en el mandato Supervisar Mensaje (MONMSG) hace que se supervisen todos los mensajes del grupo numérico. Véase el apartado " Supervisión de Mensajes en un Programa o Procedimiento CL" en el tema 8.3 para obtener más información.

7.2.2 Definición de Mensajes y Ayuda de Mensaje

Puede definir dos niveles de mensajes en el mandato ADDMSGD. El texto del mensaje es necesario y debe identificar la condición que provocó que se emitiera el mensaje. La ayuda de mensajes es opcional y debe explicar la condición con más detalla o indicar la acción de corrección que se ha de llevar a cabo. Para obtener ayuda de mensajes, el usuario de la estación de pantalla debe colocar el cursor en la línea de mensajes y pulsar la tecla Ayuda cuando se visualice el mensaje. Se puede dar formato a la ayuda de mensaje para la estación de pantalla utilizando tres caracteres de control de formato. Estos caracteres se pueden utilizar para que el usuario pueda leer mejor la ayuda de mensajes (normalmente información de ayuda en línea).

Cada uno de los tres caracteres de control de formato debe ir seguido por un espacio en blanco para separarlos del texto del mensaje.

&Nb (donde b es un blanco)
Fuerza el texto a una nueva línea (columna 2). Si el texto es más largo que una línea, las líneas siguientes se sangran de la columna 4 hasta el final del texto o hasta que se encuentre otro carácter de control de formato.

&Pb (donde b es un blanco)
Fuerza el texto a una nueva línea, sangrada hasta la columna 6. Si el texto es más largo que una línea, las líneas siguientes comienzan en la columna 4 hasta el final del texto o hasta que se encuentre otro carácter de control de formato.

&Bb (donde b es un blanco)
Fuerza el texto a una nueva línea, que comienza en la columna 4. Si el texto es más largo que una línea, las líneas siguientes se sangran de la columna 6 hasta el final del texto o hasta que se encuentre otro carácter de control del formato.

7.2.3 Asignación de un Código de Gravedad

El código de gravedad que se asigna a un mensaje en el mandato ADDMSGD indica la importancia del mensaje. Cuanto más alto es el código de gravedad, más grave es la condición. A continuación figuran los códigos de gravedad que puede utilizar y su significado. (Estos códigos de gravedad y sus significados son coherentes con los códigos de gravedad asignados a los mensajes predefinidos por IBM.)

00: Información. Para fines informativos solamente; no se ha detectado ningún error y no se precisa una respuesta. El mensaje podría indicar que hay una función en curso o que una función ha finalizado de forma satisfactoria.

10: Aviso. Existe una condición de error posible. El procedimiento o programa puede haber tomado un valor por omisión, como suministrar la entrada que falta. Se supone que el resultado de la operación es satisfactorio.

20: Error. Se ha detectado un error, pero es probablemente un error para el cual se han aplicado procedimientos de recuperación automáticos; el proceso continúa. Es posible que se haya tomado un valor por omisión para sustituir la entrada errónea. El resultado de la operación puede no ser válido. Es posible que la operación sólo se haya realizado parcialmente; por ejemplo, algunos elementos de una lista pueden haberse procesado correctamente y otros no.

30: Error grave. El error detectado es demasiado grave para que se produzca una recuperación automática, y es posible emplear valores por omisión. Si el error se halla en los datos fuente, se elude todo el registro de entrada. Si el error se ha producido durante el proceso de un procedimiento o programa, el programa o procedimiento termina de forma anómala (gravedad 40). El resultado de la operación no es válido.

40: Finalización anómala del procedimiento o función. La operación ha finalizado, probablemente debido a que el procedimiento o programa no pudo manejar datos que no eran válidos, o porque el usuario la ha cancelado.

50: Finalización anómala del trabajo. El trabajo ha finalizado o no se ha arrancado. Un paso de direccionamiento puede haber finalizado de forma anómala o no ha podido arrancar, una función a nivel de trabajo puede no haberse realizado tal como se requería o el trabajo puede haberse cancelado.

60: Estado del sistema. Emitido solamente para el operador del sistema. Proporciona el estado o un aviso sobre un dispositivo, un subsistema o el sistema.

70: Integridad del dispositivo. Emitido solamente para el operador del sistema. Indica que un dispositivo funciona mal o que de alguna forma ya no es operativo. Es posible que el usuario pueda solucionar la anomalía o tal vez sea necesaria la ayuda de un representante de servicio técnico.

80: Alerta sistema. Se emite un mensaje con un código de gravedad 80 para mensajes inmediatos. También advierte de una condición que, si bien no es lo suficientemente grave como para detener el sistema en ese preciso momento, podría llegar a serlo si no se toman las medidas oportunas.

90: Integridad del sistema. Emitido solamente para el operador del sistema. Describe una condición debido a la cual el sistema o un subsistema deja de ser operativo.

99: Acción. Se necesita algún tipo de acción manual, tal como entrar una respuesta, cambiar los formularios de la impresora o sustituir los disquetes.

Para ver información detallada del parámetro SEV, consulte el manual CL Reference.

7.2.4 Definición de Variables de Sustitución

En el parámetro FMT del mandato ADDMSGD, puede especificar variables de sustitución para los mensajes de primer o segundo nivel. Por ejemplo:

```
  Archivo &1 no encontrado
```

contiene la variable de sustitución &1. Cuando se visualiza o recupera el mensaje, la variable &1 se sustituye por el nombre del archivo que no se ha encontrado. El emisor del mensaje proporciona este nombre. Por ejemplo:

```
  Archivo ORDHDRP no encontrado
```

Compárese con:

```
  Archivo no encontrado
```

Las variables de sustitución pueden hacer que el mensaje sea más específico y que su significado sea más completo.

La variable de sustitución debe empezar con el signo & y estar seguido por una n, siendo n un número del 1 al 99. Por ejemplo, para el mensaje:

```
  Archivo &1 no encontrado
```

la variable de sustitución se define como:

```
  FMT((*CHAR 10))
```

Cuando asigna números a las variables de sustitución, debe empezar por el número 1 y utilizar números consecutivos. Por ejemplo, &1, &2, &3, y así sucesivamente. Sin embargo, no es preciso utilizar todas las variables de sustitución definidas para una descripción de mensaje en el mensaje que se envía.

Por ejemplo, el mensaje:

```
  Archivo &3 no disponible
```

es válido aunque no se utilicen &1 y &2 en los mensajes. Sin embargo, para hacer esto, debe definir &1, &2 y &3 en el parámetro FMT del mandato ADDMSGD. Para el mensaje anterior, el parámetro FMT puede ser:

```
  FMT((*CHAR 10) (*CHAR 2) (*CHAR 10))
```

donde el primer valor describe &1, el segundo &2 y el tercero &3. La descripción para &1 y &2 deben estar presentes si se utiliza &3. Además, cuando se envía este mensaje, el parámetro MSGDTA en el mandato Enviar Mensaje de Programa (SNDPGMMSG) debe incluir todos los datos descritos en el parámetro FMT. Para enviar el mensaje anterior, el parámetro MSGDTA debe tener una longitud de 22 caracteres como mínimo.

Para el mensaje anterior, también puede especificar el parámetro FMT como:

```
  FMT((*CHAR 0) (*CHAR 0) (*CHAR 10))
```

Puesto que &1 y &2 no se utilizan en el mensaje, los puede describir con una longitud 0. Entonces no es necesario enviar datos de mensaje (en este caso, el parámetro MSGDTA del mandato SNDPGMMSG debe tener 10 caracteres de longitud en este caso).

Un ejemplo de utilización de &3 en el mensaje y de la inclusión de &1 y &2 en el parámetro FMT es aquel en el que &1 y &2 se especifican en el parámetro DMPLST. (El parámetro DMPLST especifica que los datos han de volcarse cuando se envía este mensaje como un mensaje de escape a un programa que no está supervisándolo).

No es necesario especificar las variables de sustitución en el mismo orden en el que están definidas en el parámetro FMT. Por ejemplo, puede definir tres valores en el parámetro FMT como:

```
  FMT((*CHAR 10) (*CHAR 10) (*CHAR 7))
```

Las variables de sustitución pueden utilizarse en el mensaje de la forma siguiente:

```
  Objeto &1 del tipo &3 en la biblioteca &2 no está disponible
```

Si se envía este mensaje en un procedimiento o programa CL, puede concatenar los valores utilizados para los datos del mensaje, como:

```
  SNDPGMMSG .....MSGDTA(&OBJ *CAT &LIB *CAT &OBJTYPE)
```

Debe especificar el formato de los campos de datos del mensaje para la

variable de sustitución especificando el tipo de datos y, opcionalmente, la longitud en el mandato ADDMSGD. Los tipos de datos válidos para los campos de datos del mensaje son:

- Serie de caracteres entrecomillada (*QTDCHAR). Serie de caracteres que ha de colocarse entre apóstrofos. No se suprimen los blancos anteriores y los de cola. Si no especifica la longitud en la descripción del mensaje, el emisor determina la longitud del campo.

- Serie de Caracteres (*CHAR). Serie de caracteres que no debe incluirse entre apóstrofos. Se suprimen los blancos de cola. Si no especifica la longitud en la descripción del mensaje, el emisor determina la longitud del campo.

- Serie de Caracteres Convertibles (*CCHAR). Serie de caracteres que no debe incluirse entre apóstrofos. Se suprimen los blancos de cola. La longitud siempre la determina el remitente. Si se envían datos de este tipo a una cola de mensajes cuyo identificador de CCSID es distinto de 65535 ó 65534, los datos se convierten del CCSID de los datos del mensaje al CCSID de la cola de mensajes. También pueden producirse conversiones en datos de este tipo cuando se obtienen datos de la cola de mensajes utilizando una función de recepción o visualización. Consulte el manual International Application Development para obtener más información acerca de la utilización de manejadores de mensajes con los CCSID.

- Hexadecimal (*HEX). Una serie que debe ir precedida por el carácter X e incluirse entre apóstrofos; cada byte de la serie debe convertirse en dos caracteres hexadecimales (de 0 a 9 y A-F). Si no especifica la longitud en la descripción del mensaje, el emisor determina la longitud del campo.

- Binario (*BIN). Un entero binario (de 2 ó 4 bytes de longitud) con formato de entero decimal con signo. Si no especifica la longitud, se supone que es de 2.

- Decimal (*DEC). Número decimal empaquetado para asignarle formato como un número decimal con signo con una coma decimal. Debe especificar la longitud; las posiciones decimales toman el valor por omisión de 0.

- Puntero del sistema (*SYP). Puntero de 16 bytes a un objeto del sistema. En un mensaje o ayuda de mensaje, el nombre de 10 caracteres del objeto se formatea igual que los datos de tipo *CHAR.

- Puntero de espacio (*SPP). Puntero de 16 bytes a un objeto del programa. En un vuelco, los datos del objeto tiene el mismo formato que los datos de tipo *HEX. *SPP no puede utilizarse como texto de sustitución en un mensaje; sólo puede utilizarse como parte del parámetro DMPLST en el mandato ADDMSGD.

Los tipos de datos siguientes son válidos solamente en las descripciones de mensajes proporcionadas por IBM y no deben utilizarse para otros mensajes:

- Intervalo de tiempo (*ITV). Intervalo de tiempo de 8 bytes que contiene la hora ajustada al segundo entero más próximo para diversas condiciones de tiempo de espera excedido.

- Indicación de fecha y hora (*DTS). Indicación de fecha y hora del sistema de 8 bytes para la cual la fecha ha de tener el formato especificado en los valores del sistema QDATFMT y QDATSEP y el formato de la hora ha de ser hh:mm:ss.

7.2.5 Especificación de Comprobación de Validez para las Respuestas

En el mandato ADDMSGD, puede indicar el tipo de respuesta que es válida para un mensaje de consulta o de notificación. Puede especificar lo siguiente (los parámetros aparecen entre paréntesis):

- Tipo de respuesta (TYPE)
  - Decimal (*DEC)
  - Carácter (*CHAR)
  - Alfabético (*ALPHA)
  - Nombre (*NAME)
- Longitud máxima de respuesta (LEN)
  - Para decimal, 15 dígitos (9 posiciones decimales)
  - Para carácter y alfabético, 32 caracteres
  - Para nombre, 10 caracteres

  Nota: Si no especifica ninguna comprobación de validez (VALUES, RANGE, REL, SPCVAL, DFT), la longitud máxima de una respuesta es de 132 caracteres para los tipos *CHAR y *ALPHA.

- Valores que pueden utilizarse para la respuesta
  - Una lista de valores (VALUES)
  - Una lista de valores especiales (SPCVAL)
  - Un rango de valores (RANGE)
  - Una relación simple que el valor de respuesta debe cumplir (REL)

  Nota: Los valores especiales son valores que pueden aceptarse pero que no satisfacen ningún otro valor de comprobación de validez.

Cuando un usuario de la estación de pantalla entra una respuesta a un mensaje, el teclado está en minúsculas, lo cual hace que los caracteres se entren en minúsculas. Si el programa requiere que la respuesta esté en mayúsculas, puede efectuar lo siguiente:

- Utilizar el mandato SNDUSRMSG que da soporte a una opción de tabla de conversión que toma como valor por omisión la conversión de minúsculas a mayúsculas.
- Pedir al usuario de la estación de pantalla que entre los caracteres en mayúsculas especificando en el parámetro VALUES que los caracteres sólo estén en mayúsculas.
- Especificar el parámetro VALUES como mayúsculas y utilizar el parámetro SPCVAL para convertir los caracteres correspondientes de minúsculas a mayúsculas.
- Utilizar TYPE(*NAME) si todos los caracteres que se van a entrar son letras (A-Z). Los caracteres se convierten a mayúsculas antes de comprobarse.

7.2.6 Envío de un Mensaje Inmediato y Manejo de una Respuesta
En este ejemplo, el procedimiento efectúa lo siguiente:

- Envía un mensaje de consulta inmediato a QSYSOPR
- Solicita una respuesta de sí o no (Y o N)
- Comprueba que se ha entrado una respuesta válida
- Realiza un tiempo de espera excedido si el operador no responde en 120 segundos

```
            PGM
            DCL         &MSGKEY *CHAR LEN(4)
            DCL         &MSGRPY *CHAR LEN(1)
  SNDMSG:   SNDPGMMSG   MSG('.... Respuesta Y o N') TOMSGQ(QSYSOPR) +
                          MSGTYPE(*INQ) KEYVAR(&MSGKEY)
            RCVMSG      MSGTYPE(*RPY) MSGKEY(&MSGKEY) WAIT(120) +
                          MSG(&MSGRPY)
            IF          ((&MSGRPY *EQ 'Y') *OR (&MSGRPY *EQ 'y')) DO
            .
            .
            GOTO        END
            ENDDO       /* Respuesta de Y */
            IF          ((&MSGRPY *EQ 'N') *OR (&MSGRPY *EQ 'n')) DO
            .
            .
            GOTO        END
            ENDDO       /* Respuesta de  N */
            IF          (&MSGRPY *NE ' ') DO
            SNDPGMMSG   MSG('La respuesta no ha sido ni Y ni N; +
                             repita la acción') +
                          TOMSGQ(QSYSOPR)
            GOTO        SNDMSG
            ENDDO       /* Respuesta no válida */
       /* Se ha producido un tiempo excedido */
            SNDPGMMSG   MSG('No se ha recibido ninguna respuesta del +
                          mensaje anterior en 120 seg y se supone un valor +
                          de ''Y''') TOMSGQ(QSYSOPR)
            .
            .
  END:      ENDPGM
```

El mandato SNDUSRMSG no puede utilizarse en su lugar en este procedimiento, ya que no soporta una opción de tiempo de espera excedido (SNDUSRMSG espera hasta que recibe una respuesta o hasta que se cancela el trabajo).

El mandato SNDPGMMSG envía el mensaje y especifica el parámetro KEYVAR. Éste devuelve una clave de referencia del mensaje, la cual identifica de forma exclusiva dicho mensaje a fin de que la respuesta pueda coincidir correctamente con el mandato RCVMSG. El valor KEYVAR debe definirse como un campo de tipo carácter de longitud 4.

El mandato RCVMSG especifica el valor de la clave de referencia del mensaje procedente del mandato SNDPGMMSG para que el parámetro MSGKEY reciba el mensaje específico. La respuesta se devuelve en el parámetro MSG. El parámetro WAIT especifica cuánto tiempo se esperará una respuesta antes de que se produzca una situación de tiempo de espera excedido.

Cuando se recibe la respuesta, la lógica del procedimiento comprueba si se trata de un valor Y o N en mayúsculas o en minúsculas. Generalmente el operador entra un valor en minúsculas. Si el operador entra un valor que no es un espacio en blanco ni Y o N, el procedimiento envía un mensaje diferente y a continuación repite el mensaje de consulta.

Si el operador ha entrado un blanco, no se envía ninguna respuesta al procedimiento. Si se devuelve un blanco al procedimiento, se ha producido un tiempo de espera excedido (el operador no ha respondido). El procedimiento envía un mensaje al operador del sistema que indica que no se ha recibido ninguna respuesta, y que se ha tomado el valor por omisión (el valor ''Y'' se muestra como 'Y' en la cola de mensajes). Dado que el valor supuesto de 'Y' no se visualiza como respuesta, al consultar una cola de mensajes no se puede determinar si debe responderse al mensaje o si ya se ha sobrepasado el tiempo. El procedimiento no elimina un mensaje de la cola de mensajes una vez se ha enviado. El segundo mensaje debe minimizar este inconveniente y proporciona un seguimiento de comprobación sobre lo que ha sucedido.

Si ya se ha producido el tiempo excedido y el operador responde al mensaje, la respuesta se pasa por alto. No se informa al operador de que la respuesta no se ha utilizado.

Subtemas
7.2.6.1 Envío de Mensajes Inmediatos con Caracteres de Doble Byte

7.2.6.1 Envío de Mensajes Inmediatos con Caracteres de Doble Byte

Para enviar un mensaje inmediato con texto de doble byte, limite el texto a 37 caracteres de doble byte más los caracteres de control de desplazamiento. El tamaño limitado del mensaje garantiza que se visualizará correctamente.

7.2.7 Definición de Valores Por Omisión para las Respuestas

El mandato ADDMSGD le permite especificar un valor por omisión para una respuesta al mensaje. Una respuesta por omisión debe satisfacer los mismos valores de comprobación de validez que las otras respuestas para el mensaje o especificarse como un valor especial en la descripción del mensaje. Se utiliza un valor por omisión cuando un usuario ha indicado (mediante el mandato CHGMSGQ) que deben emitirse respuestas por omisión para todos los mensajes de consulta enviados a la cola de mensajes del usuario. Las respuestas por omisión también se envían cuando se suprimen los mensajes de consulta que no se han respondido. Por ejemplo, el usuario de la estación de trabajo utiliza el mandato DSPMSG para visualizar mensajes y suprime los mensajes de consulta sin respuesta pulsando F13 para eliminar todos los mensajes o F11 para eliminar uno determinado.

Las respuestas por omisión también se utilizan cuando el atributo de trabajo de INQMSGRPY está establecido en *DFT y es posible utilizarlas si está establecido en la opción *SYSRPYL. Puede utilizar la lista de respuestas del sistema para cambiar la respuesta por omisión.

7.2.8 Especificación de Manejo de Mensajes Por Omisión para Mensajes de Escape

Para cada mensaje que crea y que puede enviarse como un mensaje de escape, puede establecer una acción de manejo de mensajes por omisión para utilizarla si el mensaje, una vez enviado, no se maneja de ninguna otra forma.

Las acciones de manejo de mensajes por omisión pueden constar de:

- Un nombre de programa por omisión. Un programa a llamar que toma una acción por omisión para manejar un mensaje. Los parámetros siguientes se pasan al programa por omisión:
  - Nombre de cola de mensajes de llamada. Este parámetro es una estructura que consta de algunos campos que identifican a dónde se envió el mensaje. Consulte el manual *System API Reference*, SC41-4801, para obtener información acerca de programas de salida de manejo por omisión, y detalles de los campos en el parámetro.
  - Clave de referencia del mensaje (4 caracteres). La clave de referencia del mensaje de escape en la cola de mensajes de llamada.
- Lista de vuelco. Lista de números de campo de datos del mensaje (los mismos números que las variables de sustitución) que indican qué objetos van a volcarse.

  Además, se puede volcar:
  - Las áreas de datos de un trabajo
  - Una estructura interna de datos de máquina de un trabajo
  - Un trabajo

    Especificar una lista de vuelco de un trabajo es equivalente a especificar el mandato Visualizar Trabajo (DSPJOB) con los parámetros **JOB(*) OUTPUT(*PRINT)**.

Si no se especifica las acciones por omisión en las descripciones de mensaje, tendrá un vuelco del trabajo (como si **DSPJOB JOB(*) OUTPUT(*PRINT)** estuviese especificado).

La acción por omisión especificada en un mensaje se realiza sólo después de completarse la acción de filtraje de mensajes sin el manejo de mensajes de escape. Consulte la sección "Manejo por Omisión" en el tema 8.3.1 para obtener más información acerca del manejo de valores por omisión.

Subtemas

7.2.8.1 Ejemplo de un Programa Por Omisión

A continuación se facilita un programa por omisión de ejemplo que podría utilizarse cuando se envía un mensaje de diagnóstico seguido de un mensaje de escape. Este programa podría ser un programa CL OPM o un programa ILE que tuviera este único procedimiento CL.

```
|               PGM        PARM(&MSGQ &MRK)
|               DCL        VAR(&MRK) TYPE(*CHAR) LEN(4)
|               DCL        VAR(&MSGQ) TYPE(*CHAR) LEN(6381)
|               DCL        VAR(&QNAME) TYPE(*CHAR) LEN(4096)
|               DCL        VAR(&MODNAME) TYPE(*CHAR) LEN(10)
|               DCL        VAR(&BPGMNAME) TYPE(*CHAR) LEN(10)
|               DCL        VAR(&BLANKMRK) TYPE(*CHAR) LEN(4) VALUE(' ')
|               DCL        VAR(&DIAGMRK) TYPE(*CHAR) LEN(4) VALUE(' ')
|               DCL        VAR(&SAVEMRK) TYPE(*CHAR) LEN(4)
|               DCL        VAR(&MSGID) TYPE(*CHAR) LEN(7)
|               DCL        VAR(&MSGDTA) TYPE(*CHAR) LEN(100)
|               DCL        VAR(&MSGF) TYPE(*CHAR) LEN(10)
|               DCL        VAR(&MSGLIB) TYPE(*CHAR) LEN(10)
|               DCL        VAR(&OFFSET) TYPE(*DEC)
|               DCL        VAR(&LENGTH) TYPE(*DEC)

|               /* Comprobación para tipo de programa OPM          */

|               IF         (%SST(&MSGQ 277 1) *EQ '0') THEN(DO)
|                 CHGVAR     VAR(&QNAME) VALUE(%SST(&MSGQ 1 10))
|                 CHGVAR     VAR(&MODNAME) VALUE('*NONE')
|                 CHGVAR     VAR(&BPGMNAME) VALUE('*NONE')
|               ENDDO
|               ELSE DO
|               /* No un programa OPM; siempre utiliza el largo del */
|               /* nombre del procedimiento                        */
|                 CHGVAR     VAR(&OFFSET) VALUE(%BIN(&MSGQ 278 4))
|                 CHGVAR     VAR(&LENGTH) VALUE(%BIN(&MSGQ 282 4))
|                 CHGVAR     VAR(&QNAME) VALUE(%SST(&MSGQ &OFFSET &LENGTH))
|                 CHGVAR     VAR(&MODNAME) VALUE(%SST(&MSGQ 11 10))
|                 CHGVAR     VAR(&BPGMNAME) VALUE(%SST(&MSGQ 1 10))
|                ENDDO
|   GETNEXTMSG: CHGVAR     VAR(&SAVEMRK) VALUE(&DIAGMRK)
|               RCVMSG     PGMQ(*SAME (&QNAME &MODNAME &BPGMNAME)) +
|                            MSGTYPE(*DIAG) RMV(*NO) KEYVAR(&DIAGMRK)
|               IF         (&DIAGMRK *NE &BLANKMRK) THEN(GOTO GETNEXTMSG)
|               ELSE DO
|                   RCVMSG     PGMQ(*SAME (&QNAME &MODNAME &BPGMNAME)) +
|                              MSGKEY(&SAVEMRK) RMV(*NO) MSGDTA(&MSGDTA) +
|                              MSGID(&MSGID) MSGF(&MSGF) MSGFLIB(&MSGLIB)
|                   SNDPGMMSG  MSGID(&MSGID) MSGF(&MSGLIB/&MSGF) +
|                              MSGDTA(&MSGDTA) TOPGMQ(*PRV (&QNAME +
|                              &MODNAME &BPGMNAME))
|                 ENDDO
|               ENDPGM
```

El programa recibe todos los mensajes de diagnóstico en orden FIFO. Entonces envía el último mensaje de diagnóstico como un mensaje de escape para que el programa anterior lo pueda supervisar.

7.2.8.2 Especificación de la Opción de Alerta

En el mandato ADDMSGD, puede especificar una opción de alerta para permitir la creación de una alerta para un mensaje. Un mensaje para el cual se puede crear una alerta, puede hacer que se cree una alerta SNA y que se envíe a un punto focal de gestión de problemas. Una alerta creada para un mensaje se puede definir utilizando el mandato Añadir Descripción de Alerta (ADDALRD). Para obtener más información acerca del soporte de alertas OS/400, consulte el manual DSNX Support.

7.2.9 Ejemplo de Cómo Describir un Mensaje

En el ejemplo siguiente, el mandato ADDMSGD crea un mensaje que se va a utilizar en aplicaciones tales como la entrada de pedidos. El mensaje se emite cuando no se encuentra un número de cliente entrado en la pantalla. El mensaje es el siguiente:

```
     Número de cliente &1 no encontrado
```

El mandato ADDMSGD para este mensaje es:

```
  ADDMSGD   MSGID(USR4310) +
            MSGF(QGPL/USRMSG)  +
            MSG('Número de cliente &1 no encontrado') +
            SECLVL('Cambiar número de cliente') +
            SEV(40) +
            FMT((*CHAR 8))
```

El mensaje se añade al archivo USRMSG de la biblioteca QGPL.

Puede utilizar el mandato DSPMSGD o WRKMSGD para imprimir o visualizar las descripciones de mensajes.

El parámetro SECLVL proporciona textos muy sencillos. Para hacer que aparezca en la pantalla Información Adicional de Mensaje, especifique **SECLVL('texto de mensaje')**. El texto que especifica en este parámetro aparece en la pantalla Información Adicional de Mensaje cuando se pulsa la tecla Ayuda después de colocar el cursor en este mensaje.

7.2.10 Definición de Mensajes de Doble Byte

Para definir un mensaje con texto de doble byte, escriba un procedimiento o programa CL utilizando el mandato ADDMSGD. El mensaje definido se coloca en un archivo de mensajes y se procede al envío. Al grabar el programa, haga lo siguiente:

1. Asegúrese de que el archivo fuente que contiene el programa es un archivo de doble byte. Especifique IGCDTA(*YES) en el mandato Crear Archivo Físico Fuente (CRTSRCPF).

2. Utilice el programa de utilidad para entrada del fuente (SEU) para entrar el programa. Los mandatos CL que utilizan los caracteres de doble byte sólo pueden entrarse a través del SEU. Por esta razón, los mensajes de doble byte deben crearse en un programa CL.

3. Limite la longitud del mensaje a 37 caracteres de doble byte, de forma que pueda visualizarse o imprimirse el mensaje completo.

   Al utilizar el mandato MONMSG, limite también el parámetro de Datos de Comparación (CMPDATA) a 6 caracteres de doble byte.

4. Si el archivo de mensajes de doble byte sustituye a un archivo de mensajes alfanumérico (como los archivos de mensajes traducidos que sólo se envían a las estaciones de pantalla de doble byte), entre un mandato similar al que viene a continuación para alterar temporalmente el archivo de mensajes alfanumérico:

   ```
   OVRMSGF MSGF(QCPFMSG) TOMSGF(DBCSLIB/QCPFMSG)
   ```

   Los mensajes de doble byte pueden visualizarse solamente en las estaciones de pantalla de doble byte.

7.3 Búsquedas del Sistema en Archivos de Mensajes

El sistema utiliza los siguientes dos pasos cuando se realizan búsquedas para recuperar un mensaje de un archivo de mensajes:

1. El sistema procesa todas las alteraciones temporales en vigor para el nombre de archivo de mensajes.

   Consulte la sección "Alteración Temporal de Archivos de Mensajes" en el tema 7.3.2 para obtener más información.

2. Si el nombre de archivo de mensajes no se ha alterado temporalmente, el sistema busca el archivo de mensajes basándose en el nombre de archivo de mensajes y biblioteca especificados al utilizar el mensaje.

   Consulte la sección "Búsqueda de un Archivo de Mensajes" en el tema 7.3.1 para obtener más información.

Subtemas

7.3.1 Búsqueda de un Archivo de Mensajes

Cuando un archivo de mensajes no se ha alterado temporalmente, se utiliza el nombre de archivo de mensajes y la biblioteca especificados (cuando se envió el archivo de mensajes) para buscar el archivo de mensajes desde la que se recupera la descripción del mensaje.

Cuando se ha alterado temporalmente un nombre de archivo de mensajes pero el identificador de mensaje no consta en el archivo alterado temporalmente, también se utilizan el nombre de archivo de mensajes y la biblioteca especificados para buscar el archivo de mensajes.

La búsqueda del sistema depende de si especifica la biblioteca del archivo de mensajes como *CURLIB o *LIBL. A continuación se describe la vía de búsqueda para *CURLIB y *LIBL:

- Especificar como *CURLIB o especificar explícitamente la biblioteca del archivo de mensajes

  El sistema busca el archivo de mensajes nombrados en la biblioteca especificada o en la biblioteca actual del trabajo (*CURLIB).

- Especificar la biblioteca del archivo de mensajes como *LIBL

  El sistema busca el archivo de mensajes especificado en la lista de bibliotecas del trabajo (*LIBL). La búsqueda se detiene una vez encontrado el primer archivo de mensajes con el nombre especificado.

Si se encuentra el archivo de mensajes, pero no contiene una descripción del identificador de mensaje, se utilizan los atributos de mensaje y el texto del mensaje CPF2457 en QCPFMSG, en vez de la descripción de mensaje que falta.

Si el archivo de mensajes no se encontró, el sistema intenta recuperar el mensaje desde el archivo de mensajes que se utilizó cuando se envió.

Nota: Un archivo de mensajes puede encontrase, pero no poderse acceder debido a algún daño o a un problema de autorización.

7.3.2 Alteración Temporal de Archivos de Mensajes

Puede alterar temporalmente archivos de mensajes utilizados en un procedimiento o programa. La creación (mandato Alterar Temporalmente Archivo de Mensajes), supresión (mandato Suprimir Alteración Temporal) y visualización (mandato Visualizar Alteración Temporal) de alteraciones temporales de archivo de mensajes es similar a otros tipos de alteración temporal (consulte la publicación *Gestión de datos*). Aquí, sin embargo, sólo se altera temporalmente el nombre del archivo de mensajes, pero no los atributos, y las reglas para aplicar dichas alteraciones son ligeramente diferentes.

Para alterar temporalmente un archivo de mensajes, utilice el mandato Alterar Temporalmente Archivo de Mensajes (OVRMSGF). El archivo alterado se especifica en el parámetro MSGF y el archivo con el que se efectúa la alteración se especifica en el parámetro TOMSGF.

Por ejemplo, para alterar temporalmente QCPFMSG con un archivo de mensajes del usuario denominado USRMSGF, se utilizaría el siguiente mandato:

```
OVRMSGF MSGF(QCPFMSG) TOMSGF(USRMSGF)
```

Cuando un mensaje predefinido se recupera o se visualiza, se busca una descripción del mensaje en el archivo que sustituye temporalmente. Si dicha descripción no se encuentra en este archivo, se busca en el archivo sustituido temporalmente.

Existen varias razones para alterar temporalmente los archivos de mensaje:

- Proporcionar respuestas por omisión o listas de vuelco modificadas. Un archivo de mensajes puede crearse con descripciones de mensajes con respuestas por omisión o listas de vuelco modificadas porque las que se encuentran en las descripciones originales de mensajes no son satisfactorias. Puede establecer varios entornos operativos, cada uno con distintas respuestas por omisión.

- Modificar los niveles de gravedad de los mensajes.

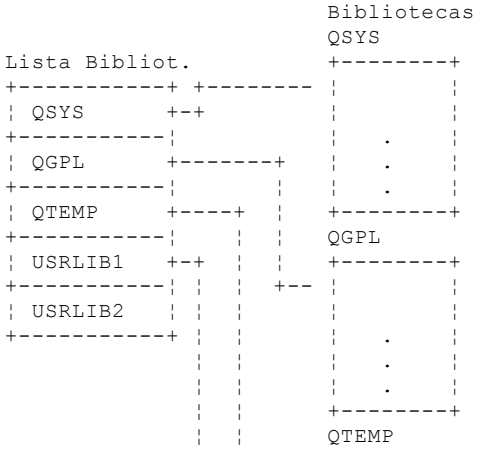
- Proporcionar un programa por omisión.

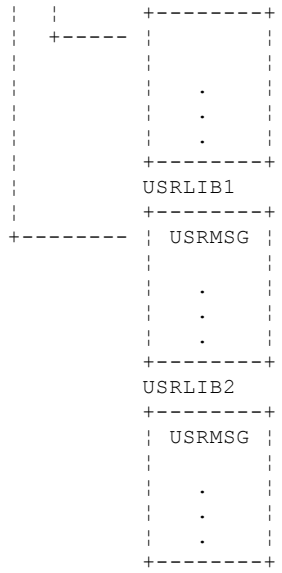
- Modificar el texto de un mensaje. Si el texto está en blanco, aparece ante el usuario como si no se hubiese enviado ningún mensaje. Por ejemplo, es posible que no quiera que el usuario vea el mensaje de estado enviado por el mandato Copiar Archivo (CPYF).

- Proporcionar la traducción de los mensajes a los idiomas nacionales. Los archivos de mensajes escritos en español pueden ser alterados temporalmente por archivos de mensajes escritos en otros idiomas. (Si se han modificado todos los mensajes, utilice la lista de bibliotecas para el trabajo para cambiar el orden de los archivos de mensajes en lugar de alterar temporalmente los archivos de mensajes.)

Otra forma de seleccionar el archivo de mensajes del que se recuperarán los mensajes es cambiando el orden de los archivos en la lista de bibliotecas para el trabajo. Sin embargo, si utiliza esta vía, la búsqueda del mensaje se detiene en el primer archivo de mensajes encontrado que tiene el nombre especificado. Si el mensaje no está en dicho archivo, la búsqueda se detiene.

Por ejemplo, supongamos que un archivo de mensajes denominado USRMSG se halla en la biblioteca USRLIB1 y otro archivo de mensajes denominado USRMSG se halla en la biblioteca USRLIB2. Para utilizar el archivo de mensajes que está en USRLIB1, USRLIB1 debe preceder a USRLIB2 en la lista de bibliotecas:





El sistema busca en el primer archivo de mensajes encontrado con el nombre correcto. Si ese archivo no contiene el mensaje, la búsqueda se detiene. Sin embargo, si utiliza el mandato OVRMSGF, el sistema busca en el archivo que lo altera temporalmente y si no encuentra el mensaje en el mismo, busca en el archivo alterado temporalmente.

Subtemas
7.3.2.1 Ejemplo de Alteración Temporal de un Archivo de Mensajes

7.3.2.1 Ejemplo de Alteración Temporal de un Archivo de Mensajes

Suponga que desea modificar un mensaje proporcionado por IBM para utilizarlo en un trabajo. Por ejemplo, imagine que desea modificar el mensaje CPC2191, cuyo contenido es el siguiente:

```
  Objeto XXX en YYY, tipo *ZZZ suprimido
```

para que contenga el siguiente texto:

```
  Objeto XXX en YYY suprimido
```

Las especificaciones sobre cómo describir el parámetro FMT se obtienen visualizando la descripción detallada de CPC2191.

En primer lugar, cree un archivo de mensajes:

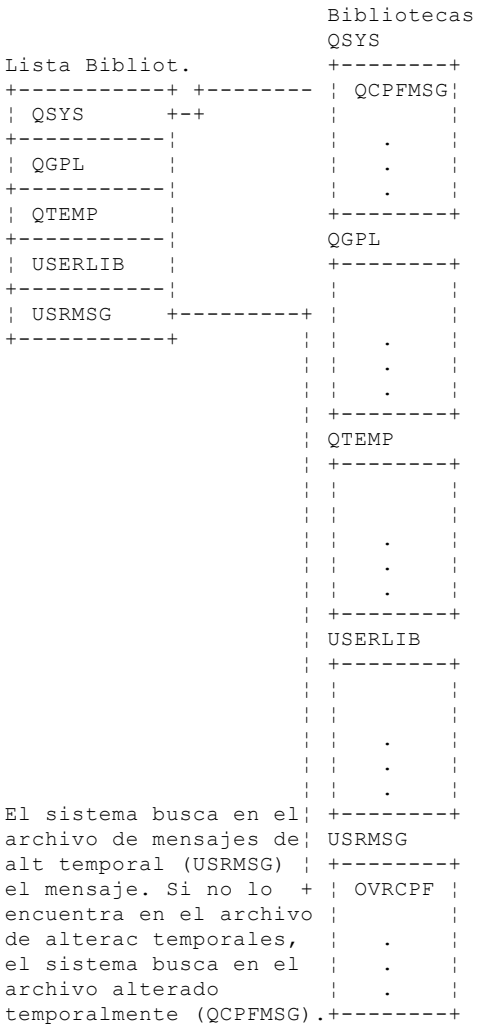
```
  CRTMSGF MSGF(USRMSG/OVRCPF)
```

Después, utilice el mensaje CPC2191 como base para su mensaje y añádalo al archivo de mensajes:

```
  ADDMSGD  MSGID(CPC2191) MSGF(USRMSG/OVRCPF) +
           MSG('Objeto &1 en &2 suprimido') +
           SEV(00) FMT((*CHAR 10) (*CHAR 10))
```

A continuación, utilice el mandato OVRMSGF para alterar temporalmente el archivo de mensajes cuando ejecute el trabajo:

```
  OVRMSGF MSGF(QCPFMSG) TOMSGF(USRMSG/OVRCPF)
```



Si desea modificar este mensaje para utilizarlo en todos sus trabajos, puede puede utilizar el mandato Cambiar Descripción del Mensaje (CHGMSGD) para cambiar el mensaje. De este modo, no tendrá que sustituir temporalmente el archivo de mensajes del sistema.

Si utiliza el mandato CHGMSGD para modificar un mensaje proporcionado por IBM, tendrá que modificar de nuevo el mensaje cuando se instale un nuevo release del sistema. Para volver a modificar el mensaje, puede realizar

los cambios en una corriente de entrada o en un programa que puede ejecutarse en cualquier momento.

También puede alterar temporalmente archivos que alteran temporalmente. Por ejemplo, puede especificar los mandatos OVRMSGF siguientes durante un trabajo.

```
  OVRMSGF  MSGF(MSGFILE1)  TOMSGF(MSGFILE2)
  OVRMSGF  MSGF(MSGFILE2)  TOMSGF(MSGFILE3)
```

Primero, el archivo MSGFILE1 se alteró temporalmente con MSGFILE2. En segundo lugar, MSGFILE2 se alteró temporalmente con MSGFILE3. Cuando se envía un mensaje, se busca en los archivos en este orden:

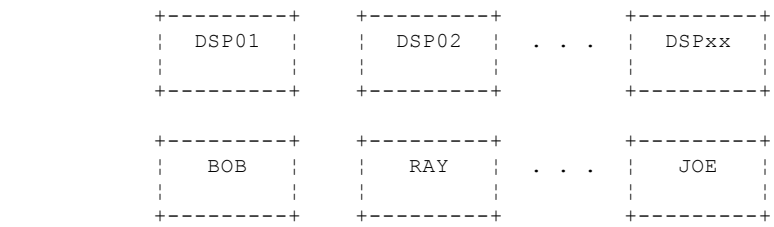
1. MSGFILE3
2. MSGFILE2
3. MSGFILE1

Puede impedir que los archivos de mensajes se alteren temporalmente. Para ello, debe especificar el parámetro SECURE en el mandato OVRMSGF.

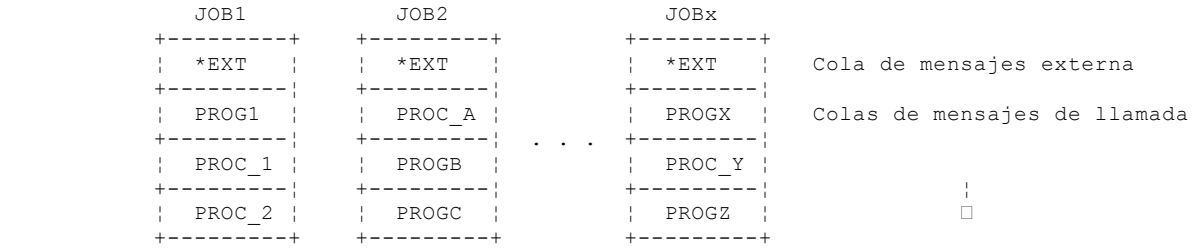
7.4 Tipos de Colas de Mensajes

Todos los mensajes del sistema se envían a una cola de mensajes. El usuario del sistema o programa del sistema asociado con la cola de mensajes recibe el mensaje procedente de la cola. De manera similar, se devuelve una respuesta a la cola de mensajes del usuario o programa que la había solicitado.

Los siguientes diagramas muestran las colas de mensajes proporcionadas por IBM. Se suministra una cola de mensajes para cada estación de pantalla (siendo DSP01 y DSP02 nombres de estaciones de pantalla) y cada perfil de usuario (siendo BOB y RAY nombres de perfil de usuario):



Se proporcionan colas de mensajes de trabajo para cada trabajo que se ejecuta en el sistema. Se concede a cada trabajo una cola de mensajes externa (*EXT) y cada llamada de un programa OPM o procedimiento ILE del trabajo tiene su propia cola de mensajes de llamada.



También se proporcionan colas de mensajes para las anotaciones históricas del sistema (QHST) y el operador del sistema (QSYSOPR):

Estas colas de mensajes se utilizan de la forma siguiente:

- La colas de mensajes de la estación de trabajo se utilizan para enviar y recibir mensajes entre los usuarios de estaciones de trabajo, así como entre usuarios de estaciones de trabajo y el operador del sistema. La cola se denomina igual que la propia estación de trabajo. El sistema crea la cola cuando se le describe a aquél la estación de trabajo.

- Las colas de mensajes de perfil de usuario pueden utilizarse para la comunicación entre usuarios. Las colas de mensajes del perfil de usuario se crean automáticamente en la biblioteca QUSRSYS cuando se crea el perfil de usuario.

- Las colas de mensajes de trabajo se utilizan para recibir solicitudes que han de procesarse (tales como los mandatos) y para enviar mensajes que resultan de procesar dichas solicitudes; los mensajes se envían al solicitante del trabajo. Existen colas de mensajes de trabajo para cada trabajo y duran únicamente lo que el propio trabajo. Las colas de mensajes de trabajo constan de una cola de mensajes externa (*EXT) y de un conjunto de colas de mensajes de entrada de pila de llamadas. Véase el apartado "Colas de Mensajes de Trabajo" en el tema 7.4.2 para obtener información.

- La cola de mensajes del operador del sistema (QSYSOPR) se utiliza para recibir y responder a los mensajes procedentes del sistema, a los usuarios de la estación de la pantalla y a los programas de aplicación.

- La cola de mensajes de anotaciones históricas se utiliza para enviar información a las anotaciones históricas (QHST) desde cualquier trabajo en el sistema.

Además de estas colas de mensajes, puede crear colas de mensajes de

usuario propias para enviar mensajes a usuarios del sistema y entre programas de aplicación.

Subtemas

7.4.1 Creación o Cambio de una Cola de Mensajes

Para crear sus propias colas de mensajes de usuario, utilice el mandato Crear Cola de Mensajes (CRTMSGQ). Además, puede utilizar el mandato Cambiar Cola de Mensajes (CHGMSGQ) para cambiar los siguientes atributos de su cola de mensajes.

Los atributos de una cola de un mensaje son los siguientes:

- Si los cambios efectuados en la cola de mensajes deben grabarse inmediatamente en el disco. La grabación inmediata de los cambios en el disco garantiza que no se perderán mensajes en situaciones como una anomalía del sistema. Observe que esto causará una disminución del rendimiento del sistema.

- El método de entrega de los mensajes que llegan a una cola de mensajes. Cuando se crea una cola de mensajes, el método de entrega se define como retener entrega. Cuando se inicia la sesión en una estación de pantalla, la cola de mensajes de usuario adopta la modalidad especificada en el perfil del usuario. Los tipos de entrega que se pueden especificar en el mandato CHGMSGQ son:

  - Entrega con interrupción. Se notifica a un trabajo se interrumpe y se llama a un programa para que entregue el mensaje. Si no hay un programa del usuario especificado en el mandato CHGMSGQ que solicita la entrega con interrupción, o si se especifica *SAME, el mandato Visualizar Mensajes visualiza automáticamente el mensaje. Los mensajes de interrupción para un trabajo pueden controlarse con el parámetro BRKMSG en el mandato CHGJOB.

  - Notificar entrega. Se notifica a un usuario de estación de pantalla que hay un mensaje en la cola mediante el indicador de Atención o una alarma sonora (o ambos). El usuario puede ver el mensaje utilizando el mandato DSPMSG.

  - Retener entrega. La cola de mensajes retiene el mensaje hasta que el usuario de la estación de pantalla lo solicita con el mandato DSPMSG.

  - Entrega por omisión. Se pasan por alto todos los mensajes, y, en el caso de aquéllos que requieren una respuesta, se envía la respuesta por omisión.

- Cómo manejar los mensajes para la entrega con interrupción.

  - Ejecute automáticamente el mandato DSPMSG. Para un trabajo interactivo, los mensajes se visualizan en la estación de pantalla si el código de gravedad es suficientemente alto. Para un trabajo por lotes, los mensajes se listan en un archivo de impresora en spool si el código de gravedad es suficientemente alto.

  - Llame a un programa manejador de interrupciones para manejar los mensajes. Debe utilizar el mandato CHGMSGQ para especificar que se ha de llamar al programa y para establecer el método de entrega en modalidad de interrupción.

- El código de gravedad para filtrar los mensajes para la entrega con interrupción y la notificación de entrega. Se visualizan los mensajes cuya gravedad es igual o mayor que el código de gravedad más bajo especificado. Cuando se crea la cola, el código de gravedad mínimo se establece en 00. Para cambiar el código de gravedad mínimo, debe utilizarse el mandato CHGMSGQ.

  Cuando se utiliza el mandato DSPMSG para visualizar mensajes que se encuentran en la cola de mensajes, puede utilizarse el parámetro de filtro de código de gravedad (SEV) para filtrar los mensajes que aparecen. Este filtro se utiliza en lugar del filtro de gravedad especificado para la cola de mensajes cuando se creó la misma. Para utilizar este filtro, especifique DSPMSG SEV(*MSGQ). Puede utilizar el mandato DSPMSG para determinar el código de gravedad actual utilizado para filtrar los mensajes con interrupción y de notificación. El código se visualiza en la línea de cabecera de la pantalla de mensajes.

- Identificador de juego de caracteres (CCSID) asociado con la cola de mensajes. Los mensajes enviados a esta cola se convierten a este CCSID. No se produce ninguna conversión si el CCSID de la cola de mensajes es 65534 ó 65535. Si el CCSID de la cola de mensajes es 65534, cada mensaje contiene su propio CCSID que establece el remitente.

- Permitir alertas para colas de mensajes estándar. Permitir alertas especifica si la cola que se crea permite que se generen alertas desde mensajes de alerta enviados a ésta.

Nota: Cuando se crea una descripción de dispositivo de estación de trabajo, el sistema establece una cola de mensajes para el dispositivo para recibir todos los mensajes de acción de este dispositivo. Para las impresoras de la estación de trabajo, unidades de cinta y dispositivos APPC, se puede utilizar el parámetro MSGQ par especificar una cola de mensaje cuando se crea una descripción de dispositivo. Si no se especifica una cola de mensaje para estos dispositivos, se utiliza el valor por omisión, QSYSOPR, como cola de mensajes. Todos los otros dispositivos son asignados a la cola de mensajes QSYSOPR cuando se crean.

La cola de mensajes definida en el perfil de usuario es conocida como una cola de mensajes de usuario. Cuando se conecta al sistema utilizando su perfil, la cola de mensajes de usuario se coloca en la modalidad de entrega especificada en el perfil de usuario.

Si su cola de mensajes de usuario está en modalidad de entrega con interrupción o de notificar entrega mientras está conectado a una estación de pantalla y se conecta a otra estación de pantalla, la cola de mensajes de usuario no cambiará la modalidad de entrega para la nueva conexión. Ningún trabajo puede modificar la modalidad de entrega de las colas de mensajes de usuario (juntamente con los mensajes de colas de trabajo y las colas de mensajes QSYSOPR) cuando la cola de mensajes está en modalidad de entrega con interrupción o de notificar entrega para otro trabajo.

Cuando se desconecta de la estación de pantalla cuando o el trabajo finaliza de forma inesperada, la modalidad de entrega de la cola de mensajes de usuario se cambia por la modalidad de retención, si la modalidad de entrega de la cola de mensajes de usuario es entrega con interrupción o notificar entrega para este trabajo. La modalidad de entrega de la cola de mensajes del usuario también se cambia de la modalidad de entrega con interrupción o notificar entrega por la modalidad retener entrega cuando se transfiere a un trabajo alternativo. Puede hacer esto utilizando el mandato Transferir Trabajo Secundario (TFRSECJOB) o pulsando la tecla Petición del Sistema y especificando la opción 1 en el menú de Petición del Sistema.

Después de la transferencia a un trabajo alternativo, se inicia la sesión utilizando su perfil de usuario. Su cola de mensajes de usuario adquiere la modalidad de entrega especificada en el perfil de usuario. Ello permite que la cola de mensajes de usuario se transfiera al trabajo alternativo. Entonces es posible establecer un intercambio de transferencias entre estos dos trabajos y tener la cola de mensajes del usuario controlada.

Sin embargo, si después de la transferencia a un trabajo alternativo, se inicia la sesión utilizando un perfil de usuario distinto del propio, la cola de mensajes del usuario para el trabajo desde el cual se ha efectuado la transferencia se deja en modalidad de retener entrega. La cola de mensajes de usuario para el perfil de usuario con el que ha iniciado la sesión se pone en la modalidad de entrega especificada en dicho perfil de usuario. Debido a esto, otro usuario puede poner su cola de mensajes de usuario en modalidad de entrega con interrupción o notificar entrega. Si otro usuario tiene todavía su cola de mensajes de usuario en dicha modalidad de entrega cuando se efectúa de nuevo la transferencia al primer trabajo, la modalidad de entrega de la cola de mensajes de usuario no puede cambiarse de nuevo a la modalidad de entrega original.

QSYSOPR es la cola de mensajes para el operador del sistema, a menos que se haya cambiado. La situación anterior puede producirse también para un operador del sistema.

Subtemas

7.4.1.1 Programa Manejador de Interrupciones

|Se llama a un programa manejador de interrupciones cuando un mensaje de
|igual o mayor gravedad que la del filtro de código de gravedad llega en
|una cola de mensajes que está en modo de entrega con interrupción. Para solicitar un programa manejador de interrupciones, debe especificar el nombre del programa y la entrega con interrupción en el mismo mandato CHGMSGQ. El programa manejador de mensajes debe recibir el mensaje con el mandato Recibir Mensaje (RCVMSG) para que el mensaje quede marcado como que ha sido manejado y no se vuelva a llamar de nuevo al programa. Para obtener más información sobre la recepción de mensajes y sobre los programas manejadores de interrupciones, véase el Capítulo 8, "Trabajar con Mensajes".

Nota: Este programa no puede abrir un archivo de pantalla si el programa interrumpido espera los datos de entrada del dispositivo de pantalla.

Puede utilizar la lista de respuestas del sistema para especificar que el sistema emita la respuesta a los mensajes de consulta predefinidos que se han especificado de forma que el usuario de la estación de pantalla no necesite responder. Véase el apartado "Utilización de la Lista de Respuestas del Sistema" en el tema 8.6 para obtener más información.

7.4.1.2 Ejemplo de Cambio de la Modalidad de Entrega

Cuando se arranca el sistema, éste coloca la cola de mensajes QSYSOPR en modalidad de entrega con interrupción cuando se arranca el subsistema de control. Sin embargo, si el operador del sistema se desconecta, la cola de mensajes vuelve a la modalidad de retener entrega. Cuando el operador del sistema se conecta de nuevo, QSYSOPR adquiere la modalidad especificada en el perfil de usuario de QSYSOPR.

El siguiente procedimiento de un programa inicial CL puede utilizarse para colocar la cola de mensajes QSYSOPR en modalidad de interrupción. Los programas iniciales pueden utilizar procedimientos similares para supervisar colas de mensajes distintas de la especificada en el propio perfil de usuario.

```
  PGM /* Procedimiento para colocar una cola mensajes en modalidad interrupción */
  CHGMSGQ  QSYSOPR DLVRY(*BREAK) SEV(50)
  MONMSG   MSGID(CPF0000) EXEC(SNDPGMMSG MSG('Imposible poner la cola +
           de mensajes QSYSOPR en modalidad *BREAK ') TOPGMQ(*EXT))
  ENDPGM
```

El procedimiento intenta establecer la cola de mensajes QSYSOPR en entrega con interrupción con un nivel de gravedad de 50. Si esto falla, se envía un mensaje a la cola externa de mensajes de trabajo (*EXT). Cuando finaliza el programa que contiene este procedimiento, se visualiza el menú inicial. El nivel de gravedad 50 se utiliza para reducir el número de mensajes de interrupción que interrumpen las tareas del usuario de la estación de trabajo. Una de las causas más comunes por las que falla es que otro usuario ya tenga QSYSOPR en modalidad de interrupción.

7.4.2 Colas de Mensajes de Trabajo

Se crean colas de mensajes de trabajo para cada trabajo del sistema con el fin de manejar todos los requisitos de mensajes de los trabajos. Las colas de mensajes de trabajo correspondientes a un solo trabajo constan de una cola de mensajes externa (*EXT) y un conjunto de colas de mensajes de llamada. A cada procedimiento ILE y programa OPM llamado dentro del trabajo se le asigna una cola de mensajes de llamada. Además, se crean unas anotaciones de trabajo para cada trabajo. Las anotaciones de trabajo son una cola lógica que conserva todos los mensajes enviados dentro de un trabajo en orden cronológico. Puede enviar mensajes a la cola *EXT o a una cola de mensajes de llamada. No envíe mensajes a las anotaciones de trabajo. El sistema también añade de forma lógica a las anotaciones de trabajo un mensaje enviado a *EXT o a una cola de mensajes de llamada.

La cola de mensajes externa (*EXT) se utiliza para comunicarse con el solicitante externo (como es el caso de un usuario de estación de pantalla) del trabajo. Los mensajes (a excepción de los mensajes de estado) enviados a la cola externa de mensajes de un trabajo también se graban en las anotaciones de trabajo (ver la sección "Anotaciones de trabajo" en el tema 8.7.1 para obtener más información).

Si un mensaje informativo, de consulta o de notificación se envía a la cola externa de mensajes para un trabajo interactivo, el mensaje se visualiza en la pantalla Visualizar Mensajes del Programa y el procedimiento espera una respuesta a los mensajes de consulta o notificación por parte del usuario de la estación de pantalla. Si el usuario no introduce una respuesta y pulsa la tecla Intro o F3 (Salir), la respuesta por omisión del mensaje se devuelve al emisor del mensaje. si no hay respuesta por omisión, se envía *N. Si se envía un mensaje de consulta o de notificación a la cola externa de mensajes para un trabajo de proceso por lotes, la respuesta por omisión se envía al emisor del mensaje. Si no hay mensaje por omisión, la respuesta es *N. La lista de respuestas del sistema puede alterar temporalmente la visualización de las consultas o el envío de respuestas por omisión a las consultas para *EXT.

Si se envía un mensaje de estado a la cola externa de mensajes de un trabajo interactivo, se visualiza el mensaje en la línea de mensajes de la estación de pantalla. Se pueden utilizar mensajes de estado de esta forma para informar al usuario de la estación de pantalla del progreso de una operación de larga ejecución. Por ejemplo, el sistema envía mensajes de estado durante la ejecución del mandato CPYF si se copia un archivo con varios miembros.

Nota: Cuando la aplicación finaliza la operación de larga ejecución, ha de enviarse otro mensaje para borrar la línea de mensajes en la pantalla. Puede utilizar para este fin CPI9801, que es un mensaje en blanco. Por ejemplo:

```
PGM
.
.
.
SNDPGMMSG  MSGID(CPF9898)  MSGF(QCPFMSG)  MSGDTA('Estado 1')  +
           TOPGMQ(*EXT)  MSGTYPE(*STATUS)
.
.
.
SNDPGMMSG  MSGID(CPF9898)  MSGF(QCPFMSG)  MSGDTA('Estado 2')  +
           TOPGMQ(*EXT)  MSGTYPE(*STATUS)
.
.
.
SNDPGMMSG  MSGID(CPI9801)  MSGF(QCPFMSG)  TOPGMQ(*EXT)  +
           MSGTYPE(*STATUS)
.
.
.
ENDPGM
```

Una cola de mensajes de llamada se utiliza para enviar mensajes entre un programa o procedimiento y otro programa o procedimiento. Siempre que un programa o procedimiento esté en la pila de llamadas (todavía no haya retornado) su cola de mensajes de llamada estará activa y podrán enviarse mensajes a ese programa o procedimiento. Una vez el programa o procedimiento ha retornado, la cola de mensajes de llamada del mismo deja de existir y ya no pueden enviarse mensajes al mismo. Los tipos de mensajes que pueden enviarse a una cola de mensajes de llamada son informativos, de petición, de terminación, de diagnóstico, de estado, de escape y de notificación.

La cola de mensajes de llamada para un programa OPM o procedimiento ILE se crea cuando se llama a ese programa o procedimiento. La cola de mensajes

de llamada está asociada exclusivamente a la entrada de la pila de llamadas en la que se ejecuta el programa o procedimiento. Una cola de mensajes de llamada se identifica de forma indirecta al identificar la entrada de la pila de llamadas. Una entrada de pila de llamadas se identifica por el nombre del programa o procedimiento que se ejecuta en esa entrada de pila de llamadas.

En el caso de un programa OPM, la entrada de pila de llamadas asociada se identifica por el nombre de programa de cómo máximo 10 caracteres. En el caso de un procedimiento ILE, la entrada de pila de llamadas asociada se identifica por un nombre de tres partes que consta del nombre de procedimiento de 256 caracteres como máximo, el nombre de módulo de cómo máximo 10 caracteres y el nombre de programa de hasta 10 caracteres. El nombre de módulo es el nombre del módulo en el que se compiló el procedimiento. El nombre del programa ILE es el nombre del programa ILE en el que se enlazó el módulo.

Al identificar la entrada de pila de llamadas para un procedimiento ILE, es suficiente especificar sólo el nombre de procedimiento. Si el nombre de procedimiento por sí mismo no identifica de forma exclusiva la entrada de pila de llamadas, también puede especificarse el nombre de módulo o el nombre de programa ILE. Si, cuando se envía un mensaje, un programa o procedimiento está en la pila de llamadas más de una vez, el nombre especificado identificará la aparición llamada más recientemente de dicho programa o procedimiento.

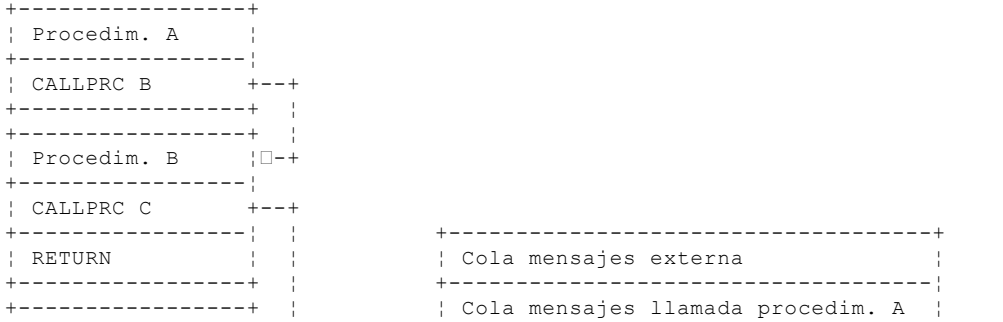
Existen otros métodos para identificar una entrada de pila de llamadas. Estos métodos se explican detalladamente en el apartado "Identificación de Entrada de Pila de Llamadas en SNDPGMMSG" en el tema 8.2.6.

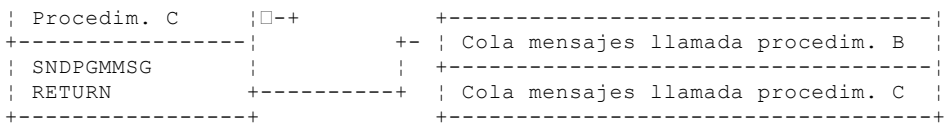
Si un programa OPM o ILE se compila y después se sustituye mientras está en la pila de llamadas, tenga cuidado al utilizar el nombre de programa para hacer referencia a una entrada de pila de llamadas. Para entradas de pila de llamadas anteriores en la pila al punto en el que se ha realizado la operación de sustitución, la referencia de nombre se resolverá con el objeto sustituido que ahora existe en QRPLOBJ. Estas referencias de nombre son válidas siempre y cuando el objeto sustituido siga existiendo en la biblioteca QRPLOBJ. Para entradas de la pila más recientes que el punto en el que se realizó la operación de sustitución, la referencia de nombre es para la nueva versión del programa. Debido a la manera en la que se determina la versión a utilizar, no sitúe un programa directamente en la biblioteca QRPLOBJ. Esta biblioteca debe utilizarse exclusivamente para la versión sustituida de un programa. Una referencia de nombre a un programa que ubique directamente en QRPLOBJ fallará.

Si se elimina un objeto de programa o se redenomina mientras una aparición del mismo está en la pila de llamadas, todas las referencias de nombre al programa eliminado y todas las referencias de nombre que utilicen el nombre antiguo fallarán. Para procedimientos ILE, si utiliza sólo el nombre de procedimiento y de módulo para una referencia, redenominar el programa no afectará a la referencia de nombre. Si también está utilizando el nombre de programa ILE, la referencia de nombre fallará.

Una cola de mensajes para una entrada de pila de llamadas de un programa o procedimiento ya no está disponible cuando finaliza el programa o procedimiento. En ese punto, solo puede hacerse referencia a un mensaje de la cola de mensajes de llamada asociada utilizando la clave de referencia de mensajes de ese mensaje.

Por ejemplo, tal como se muestra en la figura siguiente, supongamos que un procedimiento A llama a un procedimiento B, que a su vez llama a un procedimiento C. El procedimiento C envía un mensaje el procedimiento B y finaliza. El mensaje está disponible para el procedimiento B. Sin embargo, cuando finaliza el procedimiento B, su cola de mensajes de llamada ya no está disponible y el procedimiento A no puede acceder a la misma, aunque el mensaje se muestre en las anotaciones de trabajo. El Procedimiento A no puede acceder a los mensajes enviados al Procedimiento B a menos que el Procedimiento A tenga la clave de referencia de mensajes de ese mensaje.

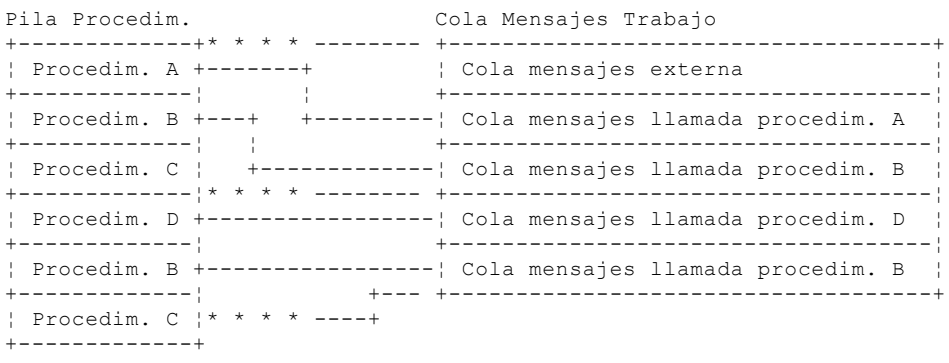




Si el procedimiento A necesita suprimir mensajes específicos, efectúe lo siguiente:

- Haga que el procedimiento C envíe mensajes específicos al procedimiento A
- Haga que el procedimiento B vuelva a enviar los mensajes al procedimiento A

La figura siguiente muestra la relación entre las llamadas de procedimiento, la cola de mensajes de trabajo y las colas de entrada de pila de llamadas. Una línea (-----) indica qué cola de mensajes está asociada con qué llamada de un procedimiento



En la figura anterior, el procedimiento B tiene dos colas de entrada de pila de llamadas, una para cada llamada del procedimiento. No hay ninguna cola de mensajes para el procedimiento C ya que no se han enviado mensajes al procedimiento C. Cuando el procedimiento C envía un mensaje al procedimiento B, el mensaje va a la cola de entrada de pila de llamadas para la última llamada del procedimiento B.

Nota: Cuando se utiliza la pantalla de entrada de mandatos, se pueden visualizar todos los mensajes enviados a la cola de mensajes de trabajo pulsando F10 (Incluye mensajes detallados). Una vez visualizados los mensajes, puede desplazarse por la pantalla utilizando las teclas de giro.

También puede visualizar los mensajes para un trabajo mediante el mandato Visualizar Anotaciones de Trabajo (DSPJOBLOG).

8.0 Capítulo 8. Trabajar con Mensajes

Este capítulo abarca algunas de las maneras en que se pueden utilizar los mensajes para la comunicación entre usuarios y programas. Los mensajes pueden enviarse:

- De un usuario del sistema a otro usuario, aunque el receptor de los mensajes no esté utilizando el sistema en ese momento
- De un programa OPM o procedimiento ILE a otro programa OPM o procedimiento ILE
- Desde un programa o procedimiento a un usuario del sistema, aunque el receptor de los mensajes no esté utilizando el sistema actualmente

Los usuarios interactivos del sistema pueden enviar solamente mensajes y respuestas inmediatos.

Los programas OPM o procedimientos ILE pueden enviar mensajes inmediatos o predefinidos con datos definidos por el usuario. Además, los programas o procedimientos pueden:

- Recibir mensajes
- Recuperar una descripción de mensaje de un archivo de mensajes, y colocarlo en una variable del programa.
- Eliminar mensajes de una cola de mensajes
- Supervisar mensajes

Subtemas

8.1 Envío de Mensajes a un Usuario del Sistema

Pueden utilizarse varios mandatos para enviar mensajes a los usuarios del sistema:

- Enviar Mensaje (SNDMSG)
- Enviar Mensaje de Interrupción (SNDBRKMSG)
- Enviar Mensaje de Programa (SNDPGMMSG)
- Enviar Mensaje de Usuario (SNDUSRMSG)

SNDPGMMSG y SNDUSRMSG sólo puede utilizarse en programas OPM o procedimientos ILE interactivos o por lotes. No se pueden entrar en una línea de mandatos. El mandato SNDMSG envía un mensaje informativo o de consulta a la cola de mensajes del operador del operador del sistema (QSYSOPR), a la cola de mensajes de la estación de pantalla o a la cola de mensajes del usuario. Puede enviar un mensaje informativo al mismo tiempo a más de una cola de mensajes. Sin embargo, sólo puede enviar un mensaje de consulta a una cola de mensajes cada vez. El mensaje se entrega según el tipo de entrega especificado para la cola de mensajes. El mensaje no interrumpe al usuario a menos que la cola de mensajes se encuentre en modalidad de interrupción.

El mandato SNDMSG siguiente es enviado por un usuario de la estación de pantalla al operador del sistema:

```
  SNDMSG   MSG('Montar cinta en dispositivo TAP1') TOUSR(*SYSOPR)
```

El mandato SNDBRKMSG envía un mensaje inmediato desde una estación de trabajo, desde un programa o desde un trabajo a una o más estaciones de pantalla para entregarlo en modalidad de interrupción, con independencia de la modalidad de entrega en la que está establecida la cola de mensajes del receptor. Este mandato puede utilizarse para enviar un mensaje solamente a colas de mensajes de estación de pantalla. Cuando envíe cualquier mensaje que requiera la atención inmediata de un usuario de la estación de pantalla, debe utilizar el mandato SNDBRKMSG. No podrá garantizar que el mensaje causará una interrupción, ya que cada trabajo tiene control mediante la utilización del parámetro BRKMSG en el mandato Cambiar Trabajo (CHGJOB).

Si envía un mensaje de consulta, puede especificar que la respuesta se envíe a una cola de mensajes diferente de la de su estación de pantalla.

El operador del sistema envía el mandato SNDBRKMSG siguiente a todas las colas de mensajes de estación de pantalla:

```
  SNDBRKMSG MSG('La desconexión del sistema tendrá lugar en 15 minutos')
            TOMSGQ(*ALLWS)
```

La desventaja de enviar este mensaje reside en que se envía a *todos* los usuarios, no sólo a los que están activos en el momento en que se envía el mensaje.

8.2 Envío de Mensajes desde un Programa CL

Utilice el mandato Enviar Mensaje de Programa (SNDPGMMSG) o Enviar Mensaje de Usuario (SNDUSRMSG) para enviar un mensaje desde un procedimiento o programa CL.

Utilizando el mandato SNDPGMMSG, puede enviar los tipos siguientes de mensajes:

- Informativo
- De consulta
- De terminación
- De diagnóstico
- De petición
- De escape
- De estado
- De notificación

Puede enviar un mensaje desde un procedimiento o programa CL a los siguientes tipos de colas:

- Cola de mensajes externa del peticionario del trabajo (vea el apartado "Colas de Mensajes de Trabajo" en el tema 7.4.2)
- Cola de mensajes de llamada de un programa o procedimiento llamado por el trabajo (vea el apartado "Colas de Mensajes de Trabajo" en el tema 7.4.2)
- Cola de mensajes del operador del sistema
- Cola de mensajes de la estación de trabajo
- Cola de mensajes del usuario

Para enviar un mensaje desde un procedimiento o programa, puede especificar lo siguiente en el mandato SNDPGMMSG:

- Identificador del mensaje o un mensaje inmediato. El identificador del mensaje es el nombre de la descripción de un mensaje predefinido.

- Archivo de mensajes. Nombre del archivo de mensajes que contiene la descripción del mensaje cuando se envía un mensaje predefinido.

- Campos de datos de mensajes. Si se envía un mensaje predefinido, estos campos contienen los valores para las variables de sustitución del mensaje. El formato de cada campo debe describirse en la descripción del mensaje. Si se envía un mensaje inmediato, no hay campos de datos de mensaje.

- Cola de mensajes o usuario que recibirá el mensaje.

- Tipo del mensaje. A continuación se indica qué tipo de mensajes pueden enviarse a qué tipos de colas (V = válido).

Tabla 8-1. Tipos de mensajes válidos para tipos de colas de mensajes

	Tipo de cola de mensajes				
Tipo de mensaje	**Externa**	**Llamada**	**QSYSOPR**	**Estación trabajo**	**Usuario**
Informativo	V	V	V	V	V
De consulta	V		V	V	V
De terminación	V	V	V	V	V
De diagnóstico	V	V	V	V	V
De petición	V	V			
De escape		V			
De estado	V	V			
De notificación	V	V			

- Identificador de juego de caracteres (CCSID). Especifica el identificador de juego de caracteres (CCSID) en el que se encuentra el mensaje o los datos de mensaje suministrados.

- Cola de mensajes de respuesta. Nombre de la cola de mensajes que recibe la respuesta a un mensaje de consulta. Por omisión, la respuesta se envía a la cola de mensajes de llamada del procedimiento

o programa que envió el mensaje de consulta.

- Nombre de la variable de clave. Nombre de la variable CL que va a contener la clave de referencia de un mensaje.

Para enviar el mensaje creado en "Ejemplo de Cómo Describir un Mensaje" en el tema 7.2.9, debería utilizar el mandato siguiente:

```
SNDPGMMSG   MSGID(USR4310) MSGF(QGPL/USRMSG) +
            MSGDTA(&CUSNO) TOPGMQ(*EXT) +
            MSGTYPE(*INFO)
```

La variable de sustitución para el mensaje es el número de cliente. Puesto que el número de cliente varía, no puede especificar el número de cliente exacto en el mensaje. En su lugar, declare una variable CL en el procedimiento o programa CL para el número de cliente (&CUSNO). A continuación especifique esta variable como el campo de datos del mensaje. Cuando se envía el mensaje, en éste aparece el valor actual de la variable:

```
Número de cliente 35500 no encontrado
```

Además, no siempre se sabe qué estación de pantalla está utilizando el procedimiento o programa, por lo que no se puede especificar la cola de mensajes de estación de pantalla a la que se enviará el mensaje (parámetro TOPGMQ); por consiguiente, se especifica la cola externa de mensajes *EXT.

Subtemas

8.2.1 Mensajes de Consulta e Informativos

Mediante el mandato SNDUSRMSG, puede enviar un mensaje de consulta o un mensaje informativo a un usuario de estación de pantalla, al operador del sistema o a una cola de mensajes definida por el usuario. Si utiliza el mandato SNDUSRMSG para enviar un mensaje de consulta al usuario, el procedimiento o programa espera una respuesta del usuario. El mensaje puede ser inmediato o predefinido. En el caso de un trabajo interactivo, el mensaje se envía por omisión al operador de estación de pantalla. En el caso de un trabajo de proceso por lotes, el mensaje se envía por omisión al operador del sistema. Para enviar un mensaje de un procedimiento o programa utilizando el mandato SNDUSRMSG, puede especificar lo siguiente en el mandato SNDUSRMSG:

- Identificador del mensaje o un mensaje inmediato. El identificador del mensaje es el nombre de la descripción de un mensaje predefinido.
- Archivo de mensajes. Nombre del archivo de mensajes que contiene la descripción del mensaje cuando se envía un mensaje predefinido.
- Campos de datos de mensajes. Si se envía un mensaje predefinido, estos campos contienen el valor para las variables de sustitución del mensaje. El formato de cada campo debe describirse en la descripción del mensaje. Si se envía un mensaje inmediato, no hay campos de datos de mensaje.
- Respuestas válidas a un mensaje de consulta.
- Valor de respuesta por omisión a un mensaje de consulta.
- Tipo del mensaje.
- Cola de mensajes a la que se enviará el mensaje.
- Respuesta del mensaje. Variable CL, si existe, que contendrá la respuesta recibida para un mensaje de consulta.
- Tabla de conversión. Tabla de conversión que se utilizará, si la hay, para convertir el valor de respuesta. Suele utilizarse para la conversión de minúsculas a mayúsculas.
- Identificador de juego de caracteres (CCSID). Especifica el identificador de juego de caracteres (CCSID) en el que se encuentra el mensaje o los datos de mensaje suministrados.

8.2.2 Mensajes de Terminación y de Diagnóstico

Mediante el mandato SNDPGMMSG, puede enviar mensajes de diagnóstico y de terminación. Puede enviar estos tipos de mensajes a cualquier cola de mensajes desde su procedimiento o programa CL. Los mensajes de diagnóstico indican al programa o procedimiento los errores detectados por el procedimiento o programa CL. Los mensajes de terminación contienen el resultado del trabajo realizado por el procedimiento o programa CL.

Generalmente se envía un mensaje de escape a la cola de mensajes del programa o procedimiento que efectúa la llamada para indicar al mismo cuál es el problema detectado o que también se han enviado mensajes de diagnóstico. Para un mensaje de terminación, normalmente no se envía un mensaje de escape porque la función solicitada ya se ha realizado.

Como ejemplo de envío de un mensaje de terminación, supongamos que el operador del sistema utiliza el menú del operador del sistema para llamar a un programa CL denominado SAVPAY para salvar ciertos objetos. El procedimiento CL contiene sólo el siguiente procedimiento, que salva los objetos y después emite el siguiente mensaje de terminación:

```
PGM
SAVOBJ OBJ(PAY1 PAY2) LIB(PAYROLL) CLEAR(*YES)
SNDPGMMSG MSG('Los objetos de la nómina se han salvado') MSGTYPE(*COMP)
ENDPGM
```

Si el mandato SAVOBJ falla, la función del procedimiento CL realiza la comprobación, y el operador del sistema tiene que visualizar los mensajes detallados para localizar el mensaje de escape específico que explica la razón de la anomalía, tal como se describe posteriormente en este capítulo. Si el mandato SAVOBJ se ejecuta satisfactoriamente, el mensaje de terminación se envía a la cola de mensajes de llamada asociada con el programa que visualiza el menú del operador del sistema.

Una de las ventajas de los mensajes de terminación es su coherencia con los mandatos proporcionados por IBM. Muchos mandatos IBM envían mensajes de terminación que indican que una acción se ha realizado satisfactoriamente. Ver el tipo de mensaje enviado a las anotaciones de trabajo puede ayudarle a analizar el problema.

8.2.3 Mensajes de Estado

Puede enviar mensajes de estado desde su procedimiento o programa CL, utilizando el mandato SNDPGMMSG, a una cola de mensajes de llamada o a la cola de mensajes externa (*EXT) para el trabajo. Cuando se envía un mensaje de estado a una cola de mensajes de llamada, el programa o procedimiento receptor puede supervisar la llegada del mensaje de estado y manejar la condición que describe. Si el programa o procedimiento receptor no supervisa el mensaje, se devuelve el control al remitente para reanudar el proceso. Véase el apartado " Supervisión de Mensajes en un Programa o Procedimiento CL" en el tema 8.3.

8.2.4 Mensajes de Escape y de Notificación

Puede enviar mensajes de escape desde su procedimiento o programa CL a la cola de mensajes de llamada del programa o procedimiento que efectúa la llamada con el mandato SNDPGMMSG. Un mensaje de escape indica al llamador que el procedimiento o programa ha finalizado de forma anómala y el motivo de esta anomalía. El llamador puede supervisar la llegada del mensaje de escape y manejar la condición que describe. Cuando el llamador maneja la condición, el control no vuelve al remitente de un mensaje de escape.

Si el llamador es otro procedimiento del mismo programa, el programa en sí no finaliza. El procedimiento al que se ha enviado el mensaje de escape puede continuar. Si el mensaje de escape se ha enviado al propio llamador del programa, todos los procedimientos activos del programa finalizan inmediatamente. Por consiguiente, el programa no puede seguir ejecutándose. Si el llamador no supervisa un mensaje de escape, se realiza la acción del sistema por omisión.

Puede enviar mensajes de notificación desde un procedimiento o programa CL a la cola de mensajes del programa o procedimiento de llamada o a la cola externa de mensajes. Un mensaje de notificación informa al llamador de una condición bajo la cual el proceso puede continuar. El procedimiento o programa de llamada puede supervisar la llegada del mensaje de notificación y manejar la condición que describe. Si el llamador es un procedimiento de Entorno de Lenguajes Integrados (ILE), el procedimiento puede manejar la condición, enviar una respuesta y después permitir que el procedimiento emisor continúe el proceso. Si el llamador no está supervisando el mensaje, se devuelve una respuesta por omisión al emisor. El remitente reanuda el proceso. Véase el apartado " Supervisión de Mensajes en un Programa o Procedimiento CL" en el tema 8.3.

Los mensajes inmediatos no se permiten como mensajes de escape y de notificación. El sistema tiene definido el mensaje CPF9898, que puede utilizarse para los mensajes de notificación y de escape inmediatos en los programas de aplicación. Por ejemplo:

```
  SNDPGMMSG MSGID(CPF9898) MSGF(QCPFMSG) MSGDTA('Condición de error') +
            MSGTYPE(*ESCAPE)
```

8.2.5 Ejemplos de Envío de Mensajes

Ejemplo 1: El siguiente procedimiento CL permite al usuario de la estación de pantalla someter un trabajo llamando a un programa CL que contiene este procedimiento en vez de entrar el mandato Someter Trabajo (SBMJOB). El procedimiento envía un mensaje de terminación cuando se ha sometido el trabajo.

```
  PGM
  SBMJOB JOB(WKLYPAY) JOBD(USERA) RQSDTA('CALL WKLY PARM(PAY1)')
| SNDPGMMSG MSG('Trabajo WKLYPAY sometido') MSGTYPE(*COMP)
  ENDPGM
```

Ejemplo 2: El siguiente procedimiento CL cambia un mensaje basándose en un parámetro recibido de un programa que se llama desde este procedimiento. El procedimiento CL envía el mensaje como un mensaje de terminación. (El campo RCDCNT está definido como caracteres en PGMA.)

```
  PGM
  DCL &RCDCNT TYPE(*CHAR) LEN(3)
  CALL PGMA PARM(&RCDCNT)
  SNDPGMMSG MSG('PGMA terminado' *BCAT &RCDCNT *BCAT +
            'registros procesados') MSGTYPE(*COMP)
  ENDPGM
```

Ejemplo 3: El siguiente procedimiento envía un mensaje que solicita al operador del sistema que cargue un formulario especial. El mandato Recibir Mensaje (RCVMSG) espera la respuesta. El operador del sistema debe entrar al menos 1 carácter como respuesta al mensaje de consulta; sin embargo, el procedimiento no utiliza el valor de respuesta.

```
  PGM
  DCL &MSGKEY TYPE(*CHAR) LEN(4)
  SNDPGMMSG   MSG('Cargar formulario especial') TOUSR(*SYSOPR) +
              KEYVAR(&MSGKEY) MSGTYPE(*INQ)
  RCVMSG MSGTYPE(*RPY) MSGKEY(&MSGKEY) WAIT(120)
  .
  .
  ENDPGM
```

Debe especificarse el parámetro WAIT en el mandato RCVMSG para que el procedimiento espere la respuesta. Si no se ha especificado el parámetro WAIT, el procedimiento continúa con la instrucción que haya a continuación del mandato RCVMSG, sin recibir la respuesta. El parámetro MSGKEY se utiliza en el mandato RCVMSG para permitir que el procedimiento reciba la respuesta a un mensaje específico. La variable &MSGKEY del mandato SNDPGMMSG se devuelve al procedimiento para utilizarla en el mandato RCVMSG.

Ejemplo 4: El siguiente procedimiento envía un mensaje al operador del sistema cuando se ejecuta en modalidad por lotes o al operador de la estación de pantalla cuando se ejecuta desde una estación de pantalla. El procedimiento acepta el valor Y o N tanto en mayúsculas como en minúsculas. (La tabla de conversión convierte los valores en minúsculas a mayúsculas (parámetro TRNTBL) para facilitar la lógica del programa.) Si el valor entrado no es uno de estos cuatro, se emite un mensaje para el operador en el que se le indica que la respuesta no es válida.

```
  PGM
  DCL &REPLY *CHAR LEN(1)
  .
  .
  SNDUSRMSG MSG('Actualice información YTD Y o N') VALUES(Y N) +
       MSGRPY(&REPLY)
  IF (&REPLY *EQ Y)
   DO
   .
   .
   ENDDO
  ELSE
   DO
   .
   .
   ENDDO
   .
   .
  ENDPGM
```

Ejemplo 5: El siguiente procedimiento utiliza el mensaje CPF9898 para enviar un mensaje de escape. El texto del mensaje es **'El procedimiento ha**

detectado una anomalía'. Los mensajes inmediatos no pueden actuar como mensajes de escape, por lo que el mensaje CPF9898 puede utilizarse con el mensaje como datos del mismo.

```
             PGM
     .
     .
  SNDPGMMSG MSGID(CPF9898) MSGF(QCPFMSG) MSGTYPE(*ESCAPE)
       MSGDTA('El procedimiento ha detectado una anomalía')
     .
     .
             ENDPGM
```

Ejemplo 6: El siguiente procedimiento permite al operador del sistema enviar un mensaje a diversas estaciones de pantalla. Cuando el operador del sistema llama al programa, este procedimiento, que está dentro del programa llamado, visualiza una solicitud en la que el operador del sistema puede entrar el tipo de mensaje a enviar y el texto del mismo. El procedimiento concatena la fecha, hora y texto del mensaje.

```
             PGM
             DCLF WSMSGD
             DCL &MSG TYPE(*CHAR) LEN(150)
             DCL &HOUR TYPE(*CHAR) LEN(2)
             DCL &MINUTE TYPE(*CHAR) LEN(2)
             DCL &MONTH TYPE(*CHAR) LEN(2)
             DCL &DAY TYPE(*CHAR) LEN(2)
             DCL &WORKHR TYPE(*DEC) LEN(2 0)
             SNDRCVF RCDFMT(PROMPT)
             IF &IN91 RETURN /* Se finalizó la petición */
             RTVSYSVAL QMONTH RTNVAR(&MONTH)
             RTVSYSVAL QDAY RTNVAR(&DAY)
             RTVSYSVAL QHOUR RTNVAR(&HOUR)
             IF (&HOUR *GT '12') DO
             CHGVAR &WORKHR &HOUR
             CHGVAR &WORKHR (&WORKHR - 12)
             CHGVAR &HOUR &WORKHR /* Cambio de la hora militar */
             ENDDO
             RTVSYSVAL QMINUTE RTNVAR(&MINUTE)
             CHGVAR    &MSG ('Desde oper. sist. ' *CAT &MONTH *CAT '/' +
                       *CAT &DAY +
                       *BCAT &HOUR *CAT ':' *CAT &MINUTE +
                       *BCAT &TEXT)
             IF (&TYPE *EQ 'B') GOTO BREAK
  NORMAL:    SNDPGMMSG MSG(&MSG) TOMSGQ(WS1 WS2 WS3)
             GOTO ENDMSG
  BREAK:     SNDBRKMSG MSG(&MSG) TOMSGQ(WS1 WS2 WS3)
  ENDMSG:    SNDPGMMSG MSG('Mensaje enviado a estaciones de pantalla') +
                       MSGTYPE(*COMP)
             ENDPGM
```

Las DDS para el archivo de pantalla, WSMSGD, utilizadas en este programa son las siguientes:

```
  |...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
       A                                      DSPSIZ(24 80)
       A          R PROMPT                    TEXT('Solicitud')
       A                                      BLINK
       A                                      CA03(91 'Devolver')
       A                                   1  2'Enviar mensajes a estaciones de trabajo'
                                               DSPATR(HI)
       A                                   3  2'TYPE'
       A             TYPE           1   1    +2VALUES('N' 'B')
       A                                      CHECK(ME)
                                              DSPATR(MDT)
       A                                     +3'(N = Sin interrupción B = Interrupción)'
       A                                   5  2'Texto'
       A             TEXT         100   1    +2LOWER
       A
       A
```

Si el operador del sistema entra lo siguiente en la solicitud:

```
  B

  Por favor, desconéctese a las 3:30
```

Se envía el siguiente mensaje de interrupción:

```
Desde oper. sist. 10/30 02:00  Por favor, desconéctese a las 3:30
```

8.2.6 Identificación de Entrada de Pila de Llamadas en SNDPGMMSG

Si el procedimiento CL va a enviar un mensaje a un programa OPM o a otro procedimiento ILE, debe identificar la entrada de pila de llamadas a la que se envía el mensaje. El mensaje se envía a la cola de mensajes de llamada de la entrada de pila de llamadas identificada.

El parámetro TOPGMQ del mandato SNDPGMMSG se utiliza para identificar la entrada de pila de llamadas a la que se envía el mensaje. La identificación de una entrada de pila de llamadas consta de las dos siguientes partes:

- Especificación de una entrada base

  La especificación TOPGMQ(*PRV *) identifica la entrada base como aquella en la que se ejecuta el procedimiento que utiliza el mandato SNDPGMMSG. El desplazamiento se especifica como una entrada anterior a esa base. Esta especificación identifica al llamador del procedimiento que utiliza el mandato.

- Especificación de desplazamiento para una entrada base

  La especificación de desplazamiento (elemento 1 de TOPGMQ) identifica si envía el mensaje a la base (*SAME) o si lo envía al llamador de la base (*PRV).

Para comprender cómo identificar la entrada base, elemento 2 de TOPGMQ, también debe comprender la pila de llamadas cuando se ejecuta un programa ILE. Se utilizan dos programas para ilustrarlo. El Programa CLPGM1 es un programa CL OPM y el Programa CLPGM2 es un programa ILE. Dado que el programa CLPGM2 es ILE, puede constar de diversos procedimientos, como: CLPROC1, CLPROC2, CLPROC3 y CLPROC4. En la ejecución, se realizan las siguientes llamadas:

- En primer lugar se llama a CLPGM1
- CLPGM1 llama a CLPGM2
- CLPGM2 llama a CLPROC1
- CLPROC1 llama a CLPROC2
- CLPROC2 llama a CLPROC3 o CLRPOC4

Consulte la Figura 8-1 para comprender la estructura de la pila de llamadas cuando CLPROC2 llama a CLPROC4. Esta figura ilustra las siguientes consideraciones:

- Hay una correspondencia biunívoca entre una entrada de la pila de llamadas y un programa OPM; para cada llamada de un programa OPM, se añade una nueva entrada a la pila de llamadas.

- Un programa ILE, como unidad, no está representado en la pila; en su lugar, cuando se llama a un programa ILE, se añade una entrada a la pila para cada procedimiento llamado en el programa. Por consiguiente, envía un mensaje a un procedimiento ILE, no a un programa ILE.

Nota: El primer procedimiento a ejecutarse cuando se llama a un programa ILE es el Procedimiento de Entrada de Programa (PEP) para el programa. En CL, el sistema genera este procedimiento (_CL_PEP) y llama al primer procedimiento que proporciona. En este ejemplo, la entrada para el PEP está entre la entrada para el programa OPM CLPGM1 y la entrada para el procedimiento CLPROC1.

A continuación se presentan diversas maneras de especificar la entrada base de la pila de llamadas.

Procedimiento Utilizando el Mandato como Base

Si el parámetro TOPGMQ especifica TOPGMQ(*SAME *) o TOPGMQ(*PRV *), la entrada para el procedimiento que utiliza el mandato SNDPGMMSG se utiliza como base. Si se ha especificado TOPGMQ(*SAME *), el procedimiento enviará un mensaje a sí mismo. Si se ha especificado TOPGMQ(*PRV *), el procedimiento enviará un mensaje al llamador.

Nota: Tenga en cuenta la siguiente información cuando un procedimiento envía un mensaje al llamador especificando TOPGMQ(*PRV *).

- Cuando CLPROC4 y CLPROC2 envían un mensaje a los llamadores, el mensaje no deja el programa que lo contiene. El mensaje se envía entre procedimientos que están dentro del mismo programa. Si el objetivo es enviar un mensaje al llamador del programa (CLPGM1 en este ejemplo), especificar TOPGMQ(*PRV *) no es lo más adecuado.
- Cuando CLPROC1 envía su mensaje de nuevo al llamador, se salta el Procedimiento de Entrada de Programa. El mensaje se envía a

CLPGM1 aunque sea el PEP quien efectúe la llamada. Cuando se especifica TOPGMQ(*PRV *), la entrada de PEP *no es visible* y no se incluye en la operación de envío. Si se especifica TOPGMQ de alguna otra forma, el PEP es *visible* para el remitente.

La Figura 8-2 ilustra los resultados cuando CLPROC1, CLPROC2 y CLPROC4 envían cada uno un mensaje al llamador de cada procedimiento.

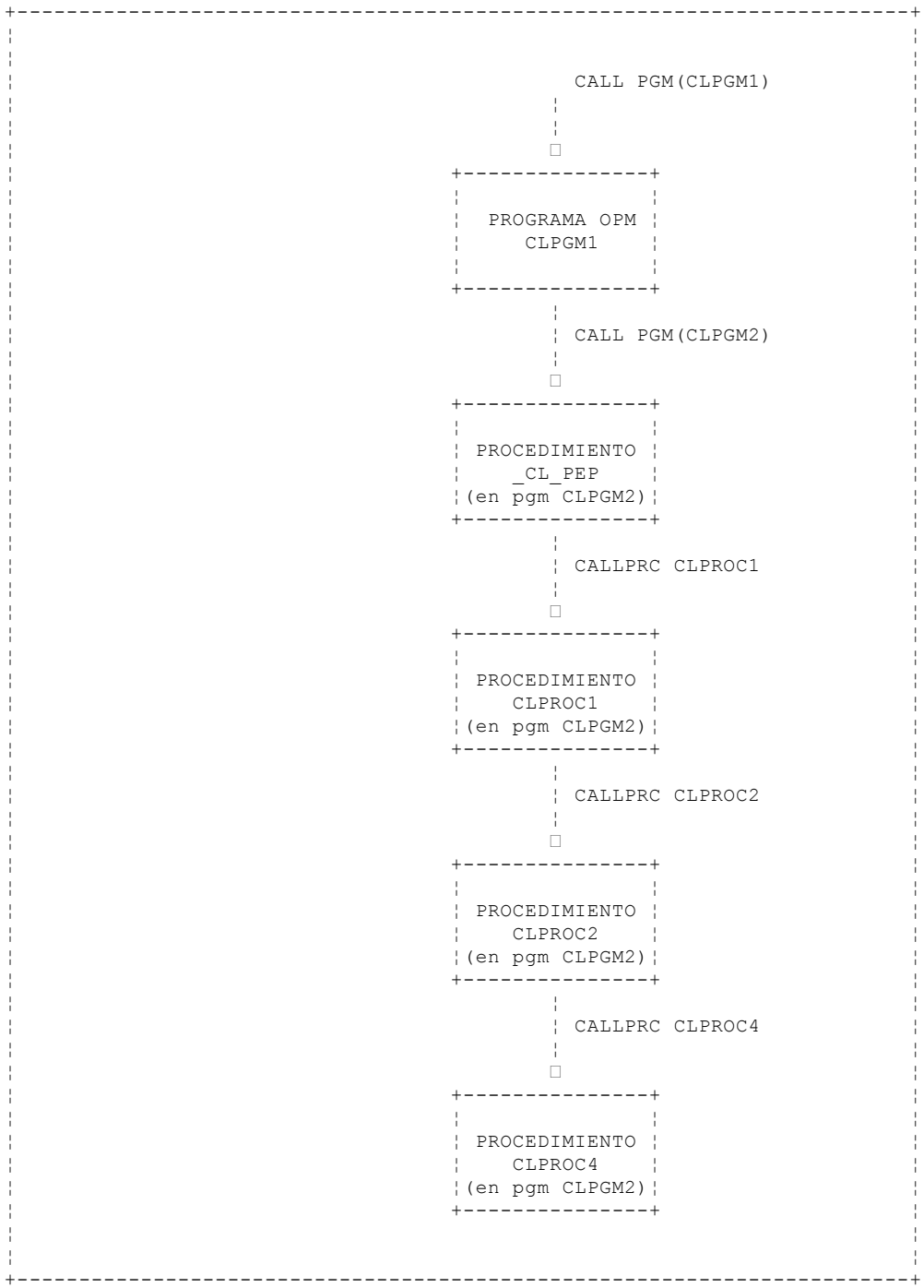
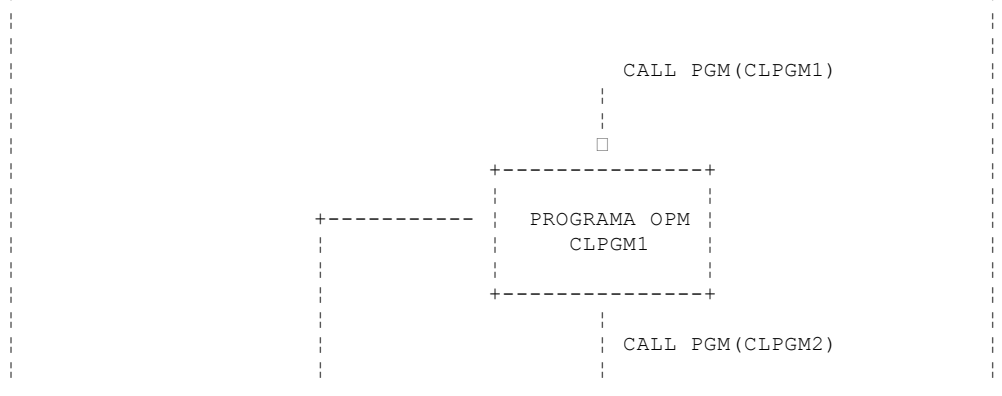


Figura 8-1. Ejemplo de pila de llamadas durante la ejecución



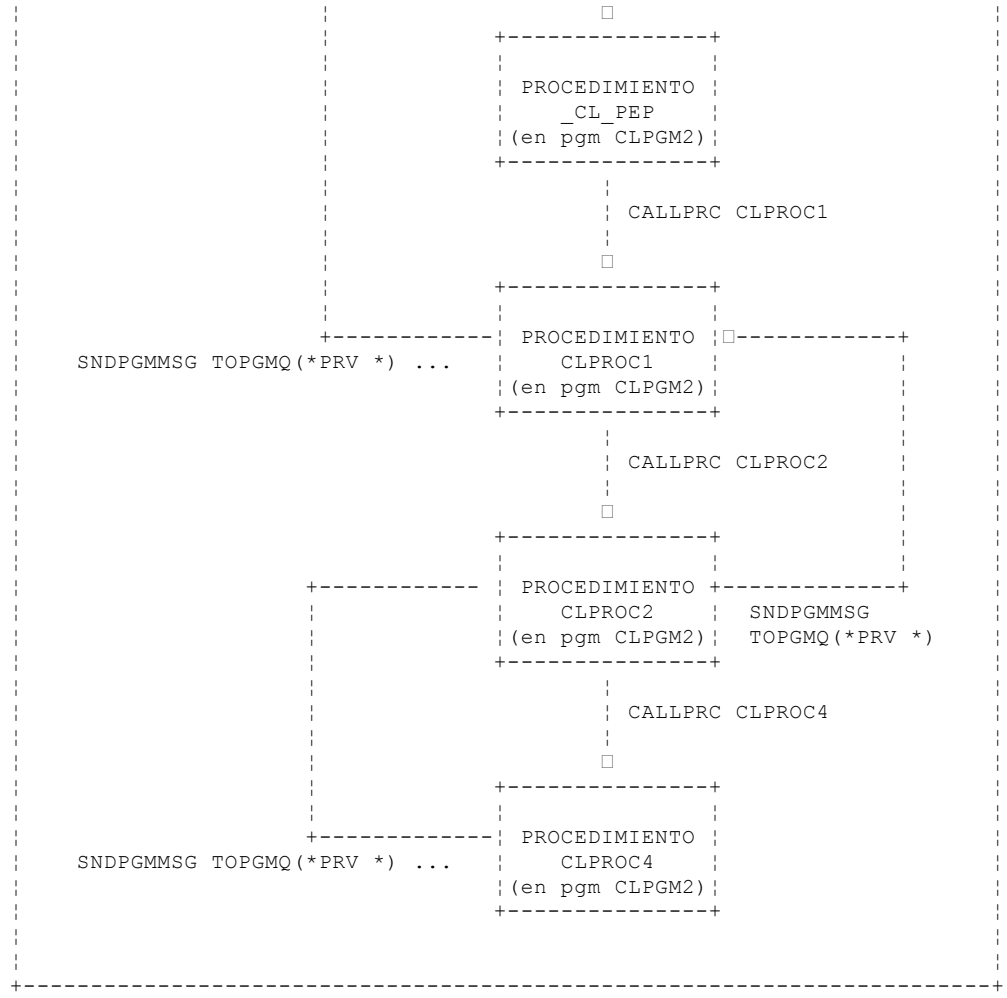


Figura 8-2. Ejemplo de TOPGMQ(*PRV *)

Identificación de la Entrada Base por Nombre

Puede identificar la entrada base proporcionando el nombre del programa OPM o procedimiento ILE en dicha entrada. El nombre proporcionado es un nombre simple (una parte) o complejo (dos o tres partes). Las siguientes son unas descripciones de nombres simples y complejos:

- Nombre simple

  Un nombre simple se utiliza para identificar un programa OPM o un procedimiento ILE. Si el nombre simple que proporciona tiene como máximo 10 caracteres de longitud, el sistema determina que el nombre es un programa OPM o un procedimiento ILE. La base se identifica como el programa OPM o procedimiento ILE llamado más recientemente por ese nombre.

  Si el nombre tiene más de 10 caracteres de longitud, el sistema determina que ese nombre corresponde a un procedimiento ILE (los nombres de programa OPM no pueden tener más de 10 caracteres). La base se identifica como la entrada para el procedimiento llamado por ese nombre más recientemente. No se tienen en cuenta las entradas que ejecutan programas OPM.

  Consulte la Figura 8-3 para ver un ejemplo de envío de un mensaje utilizando un nombre simple. En este ejemplo, CLPROC4 está enviando un mensaje a CLPROC2, y CLPROC2 está enviando un mensaje a CLPGM1.

- Nombre complejo

  Un nombre complejo consta de dos o tres partes. Éstas son:

  - nombre de módulo

    El nombre de módulo es el nombre del módulo en el que se compiló el procedimiento.
  - nombre de programa

    El nombre de programa es el nombre del programa en el que se enlazó el procedimiento.
  - nombre de procedimiento

Cuando desee identificar de forma exclusiva el procedimiento al que desea enviar el mensaje, puede utilizarse un nombre complejo en una de las siguientes combinaciones:

- nombre de procedimiento, nombre de módulo, nombre de programa
- nombre de procedimiento y nombre de módulo
- nombre de procedimiento y nombre de programa

Debe especificar el nombre de módulo como *NONE.

Si utiliza un nombre complejo, la base que se identifica no puede estar ejecutando un programa OPM.

Consulte la Figura 8-4 para ver un ejemplo de envío de un mensaje utilizando un nombre complejo. En este ejemplo, CLPROC4 está enviando un mensaje a CLPROC1 utilizando un nombre de dos partes que consta de (nombre de procedimiento, nombre de programa).

|Mas que la utilización del nombre de programa OPM completo, debe utilizar |nombres parciales. Consulte el manual CL Reference, SC41-4722 para |obtener detalles sobre el modo de partir el nombre para un mandato |específico.

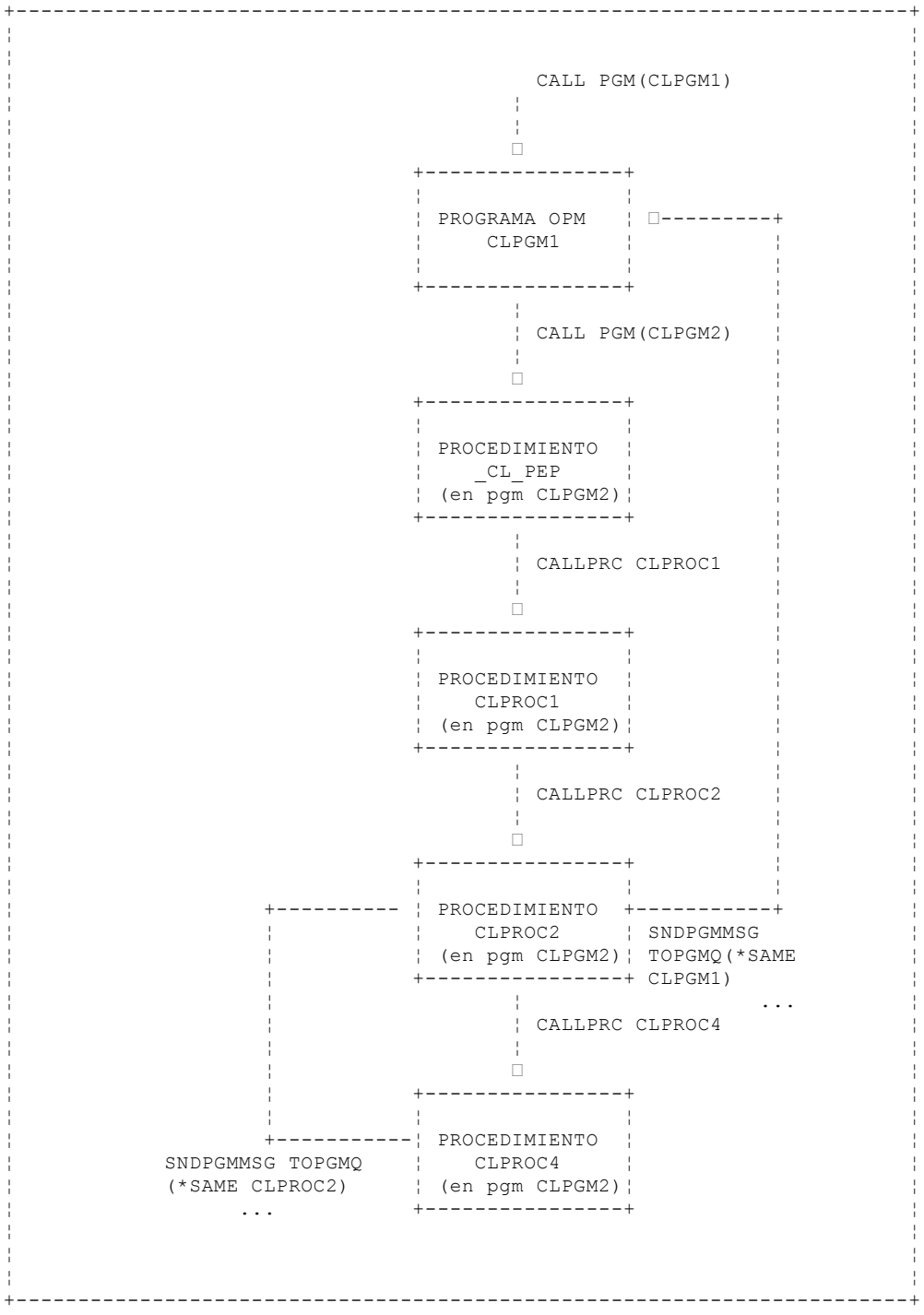


Figura 8-3. Ejemplo de utilización de un nombre simple

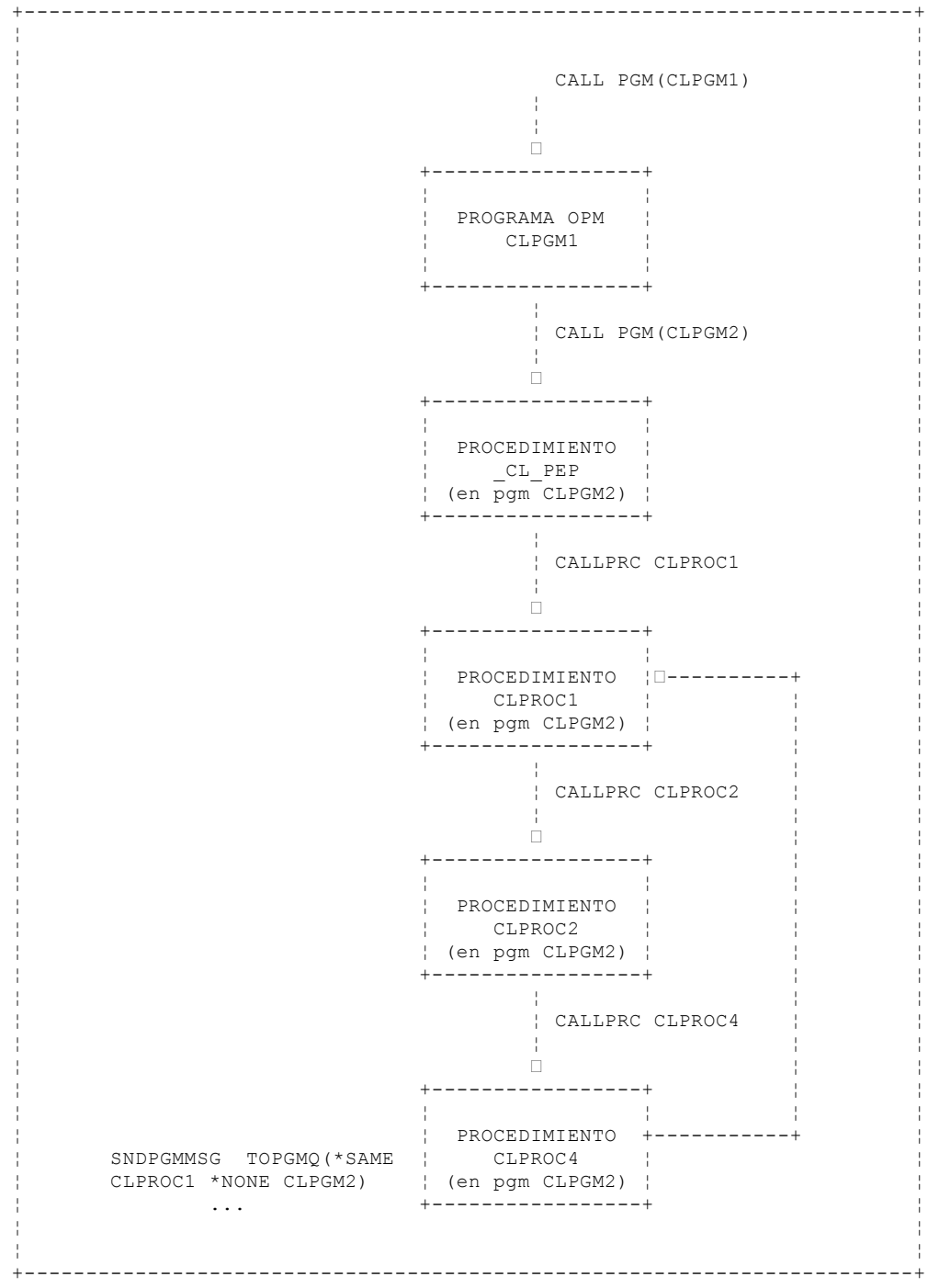


Figura 8-4. Ejemplo de utilización de un nombre complejo

Límite de Programa como Base

El valor especial *PGMBDY se utiliza por sí solo o con un nombre de programa para identificar el PEP de un programa CL. La entrada para el PEP del programa CL identificado es la entrada base. Esta opción es especialmente útil cuando desea enviar un mensaje desde dentro de un procedimiento CL fuera del límite del programa que contiene el procedimiento.

Consulte la Figura 8-5. para ver un ejemplo de envío de un mensaje utilizando el valor especial *PGMBDY. En este ejemplo, CLPROC4 está enviando un mensaje directamente a CLPGM1 que es quien efectúa la llamada del programa CLPGM2 que lo contiene. CLPROC4 puede realizar esta acción sin saber qué programa ha llamado a CLPGM2 o conociendo la ubicación del PEP en comparación con el procedimiento que envía el mensaje. En este ejemplo, *PGMBDY se utiliza sin especificar un nombre de programa que lo acompañe. Esto significa que el programa cuyo límite va a identificarse es el programa que contiene el procedimiento que está enviando el mensaje.

Consulte la Figura 8-6 para ver un ejemplo del envío de un mensaje utilizando el valor especial *PGMBDY y un nombre de programa. Los siguientes programas y procedimientos se utilizan en Figura 8-6:

- CLPGM1 y CLPGM2. Se definen como en los ejemplos anteriores.
- CLPGM3. Es el otro programa ILE

- CLPROCA en CLPGM3. Se envía un mensaje de CLPROCA al llamador de CLPGM2.

Se envía un mensaje de CLPROCA al llamador de CLPGM2 utilizando el valor especial *PGMBDY con el nombre de programa CLPGM2.

En este ejemplo, si el parámetro TOPGMQ se especifica como TOPGMQ(*PRV _CL_PEP), el mensaje se envía al llamador de CLPGM3 en vez de al llamador de CLPGM2. Esto se debe a que el procedimiento llamado más recientemente por ese nombre es el PEP para CLPGM3.

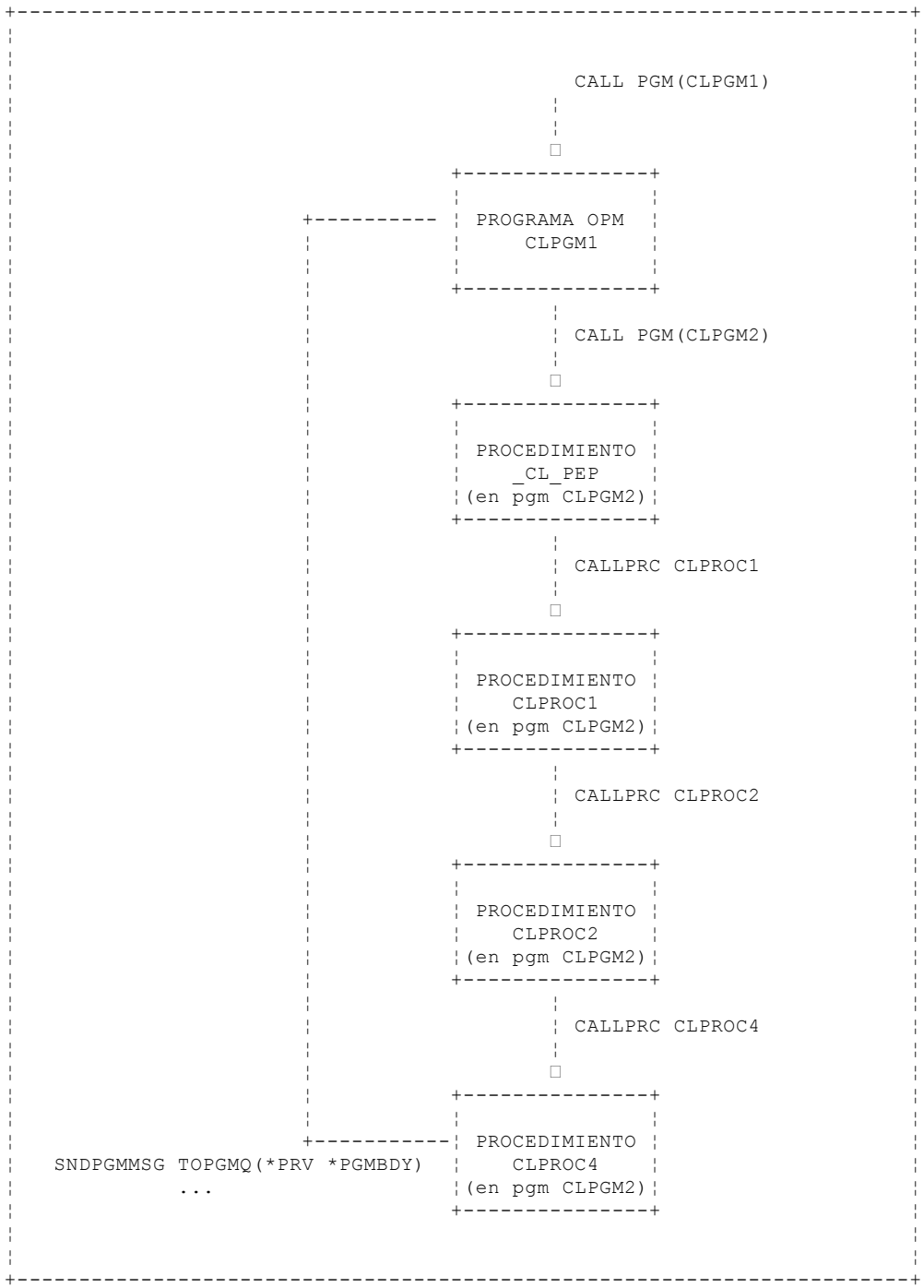
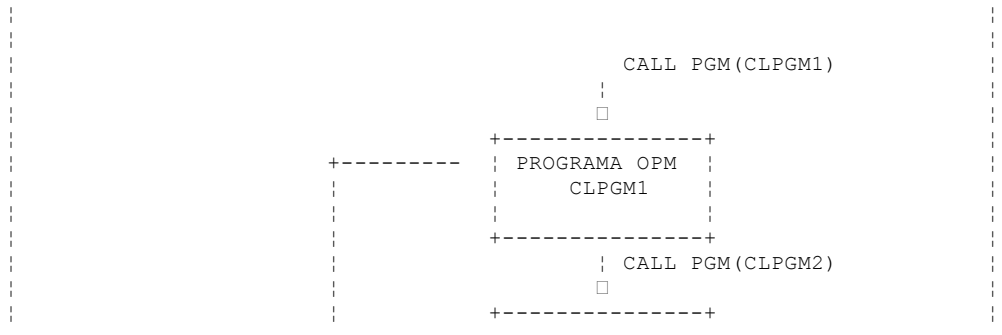


Figura 8-5. Ejemplo 1 de utilización de *PGMBDY



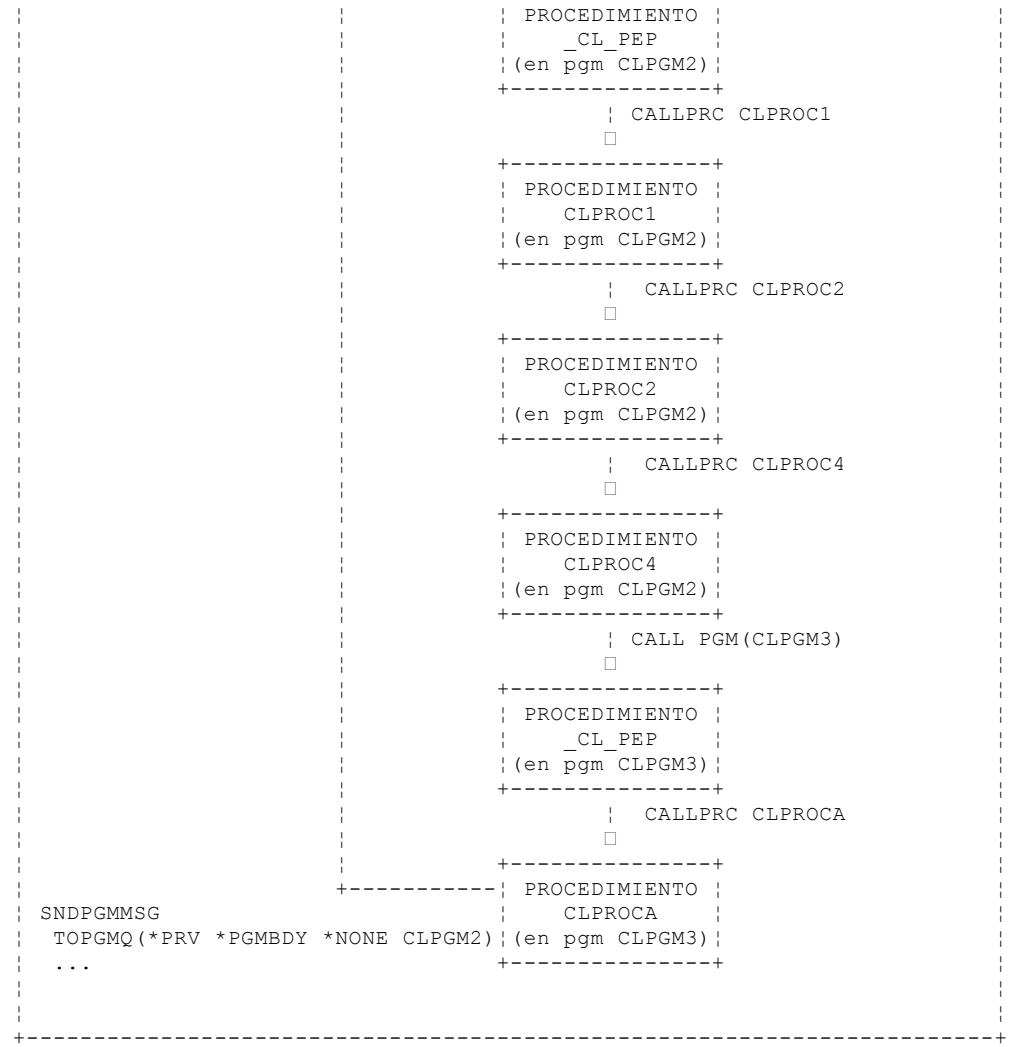
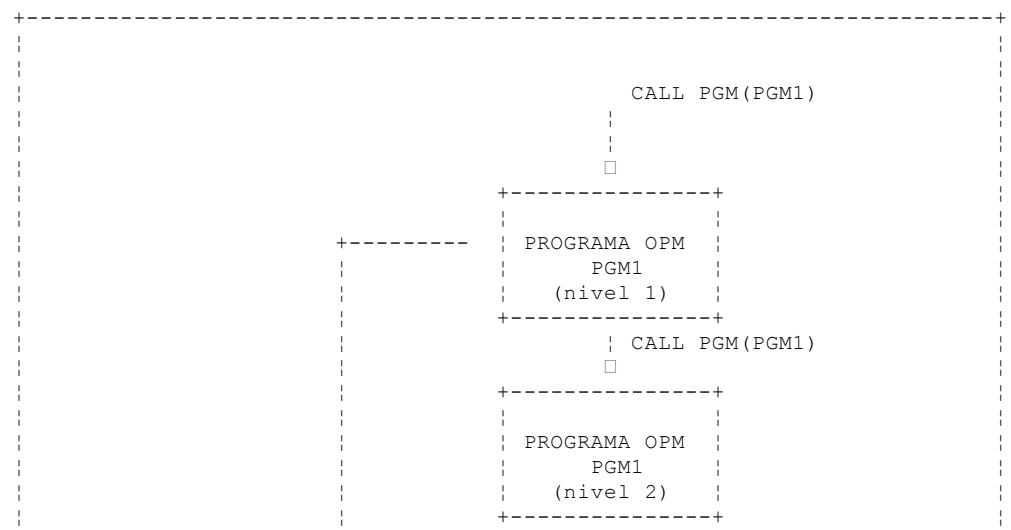


Figura 8-6. Ejemplo 2 de utilización de *PGMBDY

El valor especial *PGMBDY también se puede utilizar con un programa OPM. Si especifica un nombre de programa OPM con *PGMBDY, obtendrá los mismos resultados que cuando sólo se utiliza el nombre de programa OPM. Por ejemplo, TOPGMQ(*SAME *PGMBDY *NONE nombreopm) envía el mensaje al mismo sitio que TOPGMQ(*SAME nombreopm).

La excepción a esta regla aparece cuando se envía un mensaje a un programa OPM que se llama a sí mismo de forma recurrente. TOPGMQ(*SAME nombrepgm) envía el mensaje al último nivel de recurrencia. Sin embargo, TOPGMQ(*SAME *PGMBDY *NONE nombrepgm) envía el mensaje al primer nivel de recurrencia. La Figura 8-7 muestra cómo se llama a PGM1 y procede a llamarse a sí mismo recurrentemente dos veces más. Cuando llega al tercer nivel de recurrencia, PGM1 llama a PGM2. Entonces PGM2 envía un mensaje a PGM1. Si el programa se ha enviado utilizando sólo el nombre PGM1, el mensaje va al tercer nivel de recurrencia de PGM1. Si el programa se ha enviado utilizando el nombre PGM1 junto con el valor especial *PGMBDY, el mensaje va al primer nivel de recurrencia de PGM1.



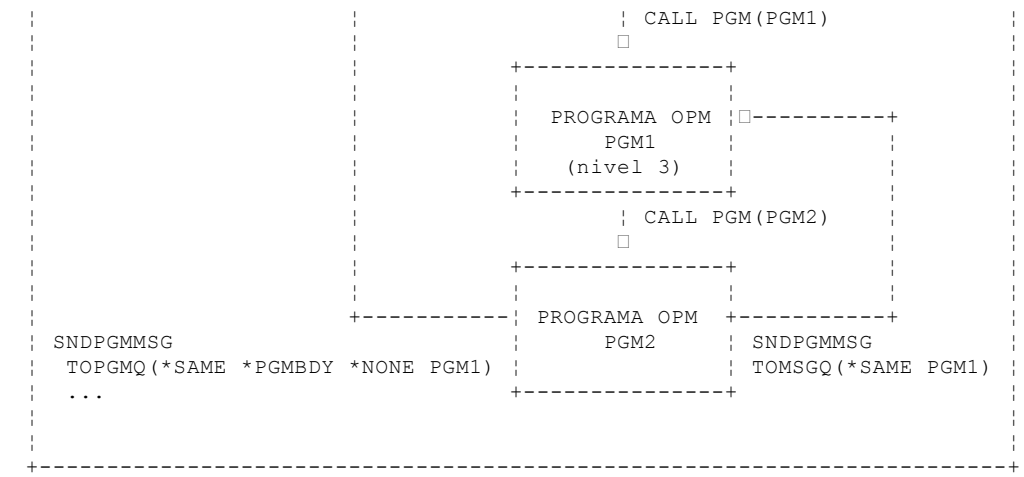


Figura 8-7. Ejemplo 3 de utilización de *PGMBDY

Procedimiento Llamado Más Recientemente como Base

Aunque no sepa el nombre de un procedimiento, puede que desee enviar un mensaje al procedimiento llamado más recientemente de un programa ILE. El valor especial *PGMNAME se utiliza con un nombre de programa ILE para utilizar el nombre de entrada base como nombre para el procedimiento llamado más recientemente del programa identificado. Los programas en este ejemplo son:

- CLPGM1 es un programa ILE con los procedimientos PROCA y PROCB.
- CLPGM2 y CLPGM3 son programas OPM.
- CLPGM3 va a enviar un mensaje a CLPGM1 y no sabe qué procedimiento es el que se ha llamado más recientemente.

El envío se efectúa utilizando el valor especial *PGMNAME y el nombre de programa CLPGM1.

Consulte la Figura 8-8. para ver un ejemplo de cómo enviar un mensaje utilizando el valor especial *PGMNAME.

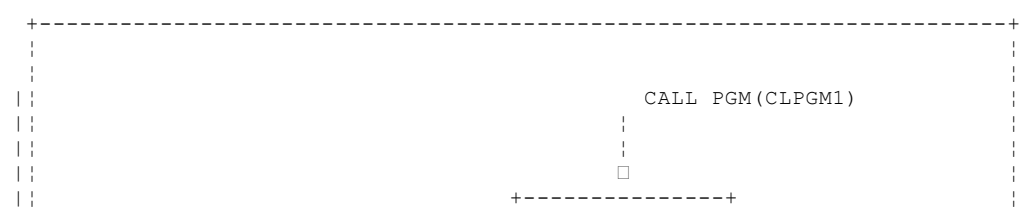
El valor especial *PGMNAME es especialmente útil si convierte algunos programas CL, pero no todos, a programas ILE. Por ejemplo, CLPGM1 es un programa CL OPM; CLPGM3 ha enviado mensajes a CLPGM1 y especifica TOPGMQ(*SAME CLPGM1). Si CLPGM1 se convierte a ILE, solo funciona el mandato SNDPGMMSG en CLPGM3 (OPM). CLPGM1 no funciona debido a que no había entrada en la pila de llamada para CLPGM1. Si cambia el mandato a TOPGMQ(*SAME *PGMNAME *NONE CLPGM1), CLPGM3 envía mensajes satisfactoriamente a CLPGM1, indiferentemente de los nombres que haya utilizado para los nombres de procedimiento.

El valor especial *PGMNAME también puede utilizarse con un nombre de programa OPM. En este caso el resultado es el mismo que si sólo utilizara el nombre. Por ejemplo, TOPGMQ(*SAME *PGMNAME *NONE opmpgm) envía el mensaje al mismo sitio que TOPGMQ(*SAME opmpgm). La utilización de *PGMNAME debería considerarse cuando no se pueda determinar si el mensaje se ha enviado a un nombre de programa OPM o a un nombre de programa ILE.

Utilización de un Límite de Control como Base

Puede identificar la entrada base como la que se encuentra en el límite de control más cercano utilizando el valor especial *CTLBDY. Un límite de control existe entre dos entradas de la pila de llamadas si las dos entradas se ejecutan en dos grupos de activación distintos. La identificada mediante la utilización de este valor especial se está ejecutando en el mismo grupo de activación que la entrada que envía el mensaje.

Consulte la Figura 8-9. para ver un ejemplo de cómo enviar un mensaje utilizando el valor especial *CTLBDY. Los tres programas de este ejemplo (CLPGM1, CLPGM2 y CLPGM3) son programas ILE. CLPGM1 se ejecuta en el grupo de activación AG1, mientras que CLPGM2 y CLPGM3 se ejecutan en el grupo de activación AG2. En este ejemplo, PROC3A envía un mensaje a la entrada inmediatamente anterior al límite para AG2.



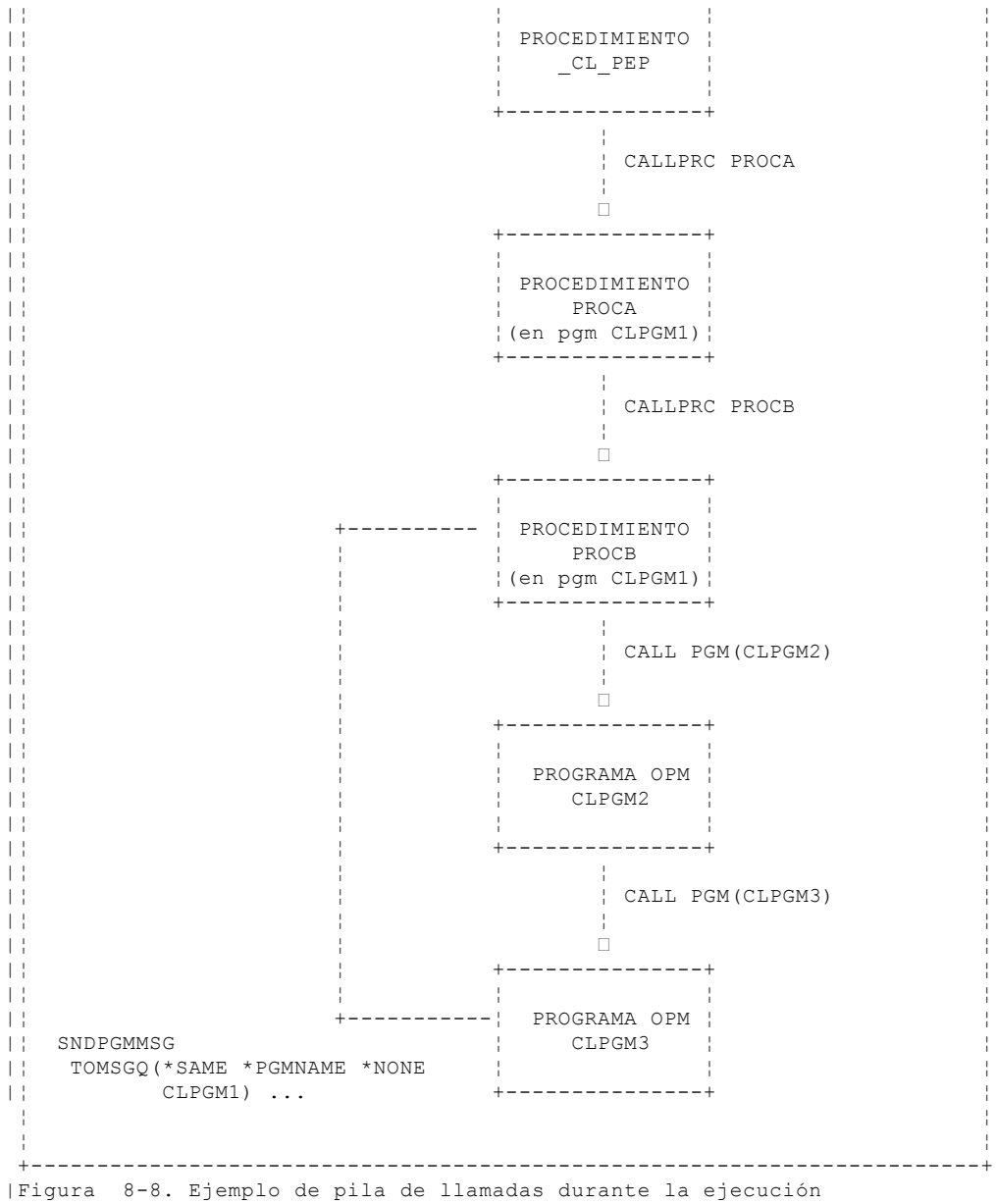
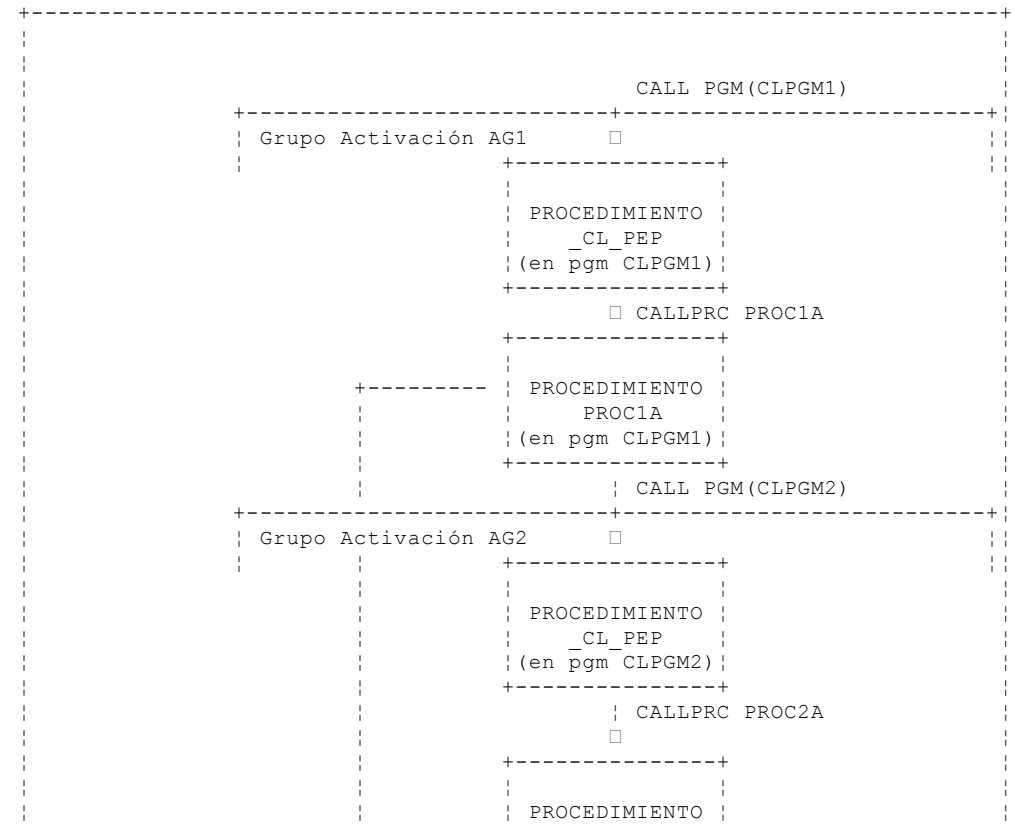


Figura 8-8. Ejemplo de pila de llamadas durante la ejecución



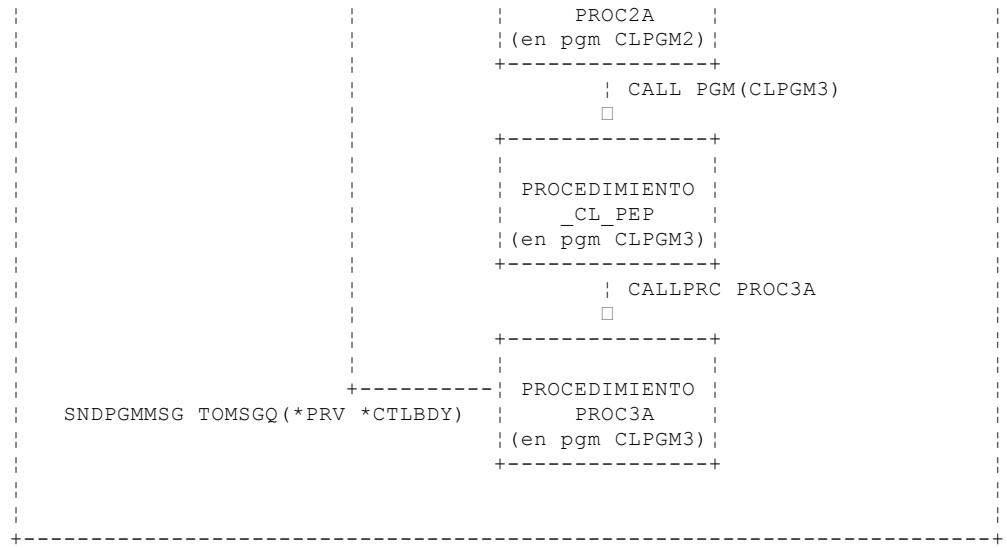


Figura 8-9. Ejemplo de utilización de *CTLBDY

Consideraciones para los Programas de Servicio

Las explicaciones anteriores se aplican a programas ILE y a programas de servicio ILE. La diferencia más importante entre un programa ILE y un programa de servicio ILE está relacionada con el manejo de mensajes. El programa de servicio no tiene un PEP.

El PEP no es necesario para ninguna de las opciones utilizadas para identificar una entrada base. La excepción a esta norma la encontramos cuando se utiliza explícitamente el nombre _CL_PEP. Por ejemplo, TOPGMQ(*PRV *PGMBDY) siempre envía un mensaje al llamador del programa o programa de servicio ILE. Si éste es un programa ILE, el PEP se identifica como la base el valor *PGMBDY identifica al PEP como la base. Si es un programa de servicio ILE, el valor *PGMBDY identifica la entrada para el primer procedimiento llamado en el programa de servicio.

8.2.7 Recepción de Mensajes en un Procedimiento o programa CL

Utilice el mandato Recibir Mensaje (RCVMSG) para recibir mensajes desde una cola de Mensajes para su procedimiento o programa. Los mensajes pueden recibirse de las siguientes maneras:

- Por tipo de mensaje. Puede especificar la recepción de todos los tipos o de un tipo determinado de mensaje (parámetro MSGTYPE). Los mensajes nuevos (aquellos que no se han recibido en el procedimiento o programa) se reciben en orden primero en entrar, primero en salir (FIFO). Sin embargo, los mensajes de tipo ESCAPE se reciben por orden de último en entrar primero en salir (LIFO).

- Por la clave de referencia del mensaje. Puede efectuar una de estas acciones:

  - Recibir un mensaje utilizando su clave de referencia de mensaje. El sistema asigna una clave de referencia a cada mensaje de una cola de mensajes y pasa la clave como datos de variable, ya que no puede imprimirse. Debe declarar esta variable en su procedimiento o programa CL (mandato DCL).Debe especificar en el mandato RCVMSG la variable CL mediante la cual se pasará la clave (parámetro MSGKEY).

  - Recibir el mensaje que se encuentra en la cola de mensajes a continuación del mensaje cuya clave de referencia se ha especificado. Además de especificar el parámetro MSGKEY, debe especificarse MSGTYPE(*NEXT).

  - Recibir el mensaje de una cola de mensajes que se encuentra delante del mensaje cuya clave de referencia se ha especificado. Además de especificar el parámetro MSGKEY, debe especificarse MSGTYPE(*PRV).

- Por su ubicación en la cola de mensajes. Debe especificarse MSGTYPE(*FIRST) para el primer mensaje de la cola de mensajes y MSGTYPE(*LAST) para el último.

- Tanto por el tipo de mensaje como por la clave de referencia de mensaje (parámetros MSGTYPE y MSGKEY).

Para recibir un mensaje, puede especificar:

- La cola de mensajes. De dónde se recibirá el mensaje.

- El tipo del mensaje. Puede especificar un tipo de mensaje concreto o todos los tipos.

- Si se ha de esperar a que llegue un mensaje. Cuando se acaba la espera y no se ha recibido mensaje alguno, las variables CL cuya devolución se ha solicitado, se rellenan con espacios en blanco (o ceros si se tratan de variables numéricas) y el control vuelve al procedimiento o programa que ejecuta el mandato RCVMSG.

- Si ha de eliminarse o no el mensaje de la cola de mensajes después de recibirlo. Si no se elimina, se convierte en un mensaje antiguo de la cola de mensajes y sólo puede recibirse de nuevo (mediante un procedimiento) a través de su clave de referencia de mensaje. Sin embargo, si los mensajes de la cola de mensajes vuelven a restaurarse como nuevos mediante el mandato CHGMSGQ, no será necesario utilizar la clave de referencia del mensaje para recibirlo. Observe que los mensajes de consulta a los que ya se ha contestado no adquieren el estado de mensaje nuevo (véase el apartado "Eliminación de Mensajes de una Cola de Mensajes" en el tema 8.2.9 para obtener más información).

- CCSID al que convertir. Especifica el CCSID en el que desea que se devuelva su texto de mensaje.

- Un grupo de variables CL en las cuales se coloca la siguiente información (cada punto corresponde a una variable):

  - Clave de referencia del mensaje en la cola de mensajes (variable de tipo carácter, 4 caracteres)
  - Mensaje (variable de tipo carácter, longitud variable)
  - Longitud del mensaje, incluida la longitud de los datos de la variable de sustitución (variable decimal, 5 posiciones decimales)
  - Información de ayuda de mensaje en línea (variable de tipo carácter, longitud variable)
  - Longitud de la ayuda del mensaje, incluida la longitud de los datos de las variables de sustitución (variable decimal, 5 posiciones decimales)

- Datos del mensaje para las variables de sustitución facilitadas por el emisor del mensaje (variable de tipo carácter, longitud variable)
- Longitud de los datos del mensaje (variable decimal, 5 posiciones decimales)
- Identificador del mensaje (variable de tipo carácter, 7 caracteres)
- Código de gravedad (variable decimal, 2 caracteres de longitud)
- Emisor del mensaje (la variable de tipo carácter, mínimo de 80 caracteres)
- Tipo de mensaje recibido (variable de tipo carácter, 2 caracteres de longitud)
- Opción de alerta del mensaje recibido (variable de tipo carácter, 9 caracteres)
- Archivo de mensajes que contiene el mensaje predefinido (variable de tipo carácter, 10 caracteres)
- Nombre de la biblioteca de archivos de mensajes que contiene el archivo de mensajes utilizado para recibir el mensaje (variable de tipo carácter, 10 caracteres)
- Nombre de la biblioteca de archivos de mensajes que contiene el archivo de mensajes utilizado para enviar el mensaje (variable de tipo carácter, 10 caracteres)
- El CCSID de los datos del mensaje es el identificador de juego de caracteres asociado a los datos de sustitución devueltos (variable decimal, 5 posiciones decimales)
- El CCSID de los datos de texto es el identificador de juego de caracteres asociado al texto devuelto por los parámetros Mensaje y Ayuda de mensaje (variable decimal, 5 posiciones decimales)

```
  RCVMSG MSGQ(QGPL/INVN) MSGTYPE(*ANY) MSG(&MSG)
```

El mensaje recibido se coloca en la variable &MSG. *ANY es el valor por omisión del parámetro MSGTYPE.

Al trabajar con la cola de mensajes de entrada de pila de llamada de un procedimiento ILE escrito en un lenguaje diferente de CL, es posible recibir un mensaje de excepción (Escape o Notificado) cuando la excepción no está todavía manejada. El mandato RCVMSG puede utilizarse para recibir un mensaje e indicar al sistema que se ha manejado la excepción.

Esto puede controlarse utilizando la palabra clave RMV. Si se especifica *NO para esta palabra clave, la excepción se maneja y el mensaje se deja en la cola de mensajes como mensaje antiguo. Si se especifica *KEEPEXCP, la excepción no se maneja y el mensaje queda en la cola de mensajes como mensaje nuevo. Si se especifica *YES, el mensaje de excepción se maneja y el mensaje se elimina de la cola de mensajes.

Puede utilizarse la palabra clave RTNTYPE para determinar si el mensaje recibido es un mensaje de excepción, y, si es así, si la excepción se ha manejado.

Subtemas
8.2.7.1 Mensajes de Petición
8.2.7.2 Escritura de Procedimientos y programas de Proceso de Peticiones
8.2.7.3 Determinación de la existencia de un Procesador de Peticiones

8.2.7.1 Mensajes de Petición

La recepción de mensajes de petición es un método para su procedimiento o programa CL para procesar mandatos CL. Por ejemplo, su procedimiento o programa puede obtener entrada de una estación de pantalla y manejar los mensajes resultantes del análisis y proceso del programa. Normalmente, los mensajes de petición se reciben de la cola externa de mensajes (*EXT) del trabajo. En el caso de trabajos por lotes, las peticiones recibidas son las que se han leído de la corriente de entrada. En el caso de trabajos interactivos, las peticiones recibidas son las que el usuario de la estación de pantalla ha entrado de una en una en la pantalla Entrada de Mandatos. Por ejemplo, los mandatos CL son peticiones que son recibidas por el procesador CL proporcionado por IBM.

Su procedimiento o programa debe definir la sintaxis de los datos del mensaje de petición, interpretar la petición y diagnosticar los posibles errores. Mientras se analiza la petición o se ejecuta la función de petición, pueden detectarse cualquier número de errores. Como resultado de estos errores, los mensajes se envían a la cola de mensajes de llamada para el procedimiento o programa. El procedimiento o programa maneja estos mensajes y después recibe el siguiente mensaje de petición. Así, se define un ciclo de proceso de peticiones; se recibe un mensaje de petición, se analiza la petición, su procedimiento o programa la ejecuta visualizando los mensajes resultantes, y se recibe la siguiente petición. Si no hay más mensajes de petición para recibir en un trabajo de proceso por lotes, se envía un mensaje de escape al procedimiento o programa para indicarlo.

Más de un programa OPM o procedimiento ILE de un trabajo pueden recibir mensajes de petición para procesarlos. Las peticiones recibidas por las llamadas de programa más recientes se consideran jerarquizadas en las recibidas por las llamadas de programa de un nivel superior. Los ciclos de proceso de peticiones en cada nivel de jerarquía son independientes entre sí. Dentro de un programa ILE, pueden haber uno o más procedimientos que reciban mensajes de petición. Si hay más de un procedimiento que procesa peticiones, se produce el anidamiento dentro del mismo programa ILE y los niveles de anidamiento permanecen independientes.

El diagrama siguiente muestra cómo QCMD procesa los mensajes de petición:

IMAGEN 8

1 El procesador CL QCMD recibe un mensaje de petición desde *EXT.

2 Si no hay ningún mensaje de petición en *EXT, se visualiza la pantalla de Entrada de Mandatos. El usuario de la estación de pantalla entra un mandato en la pantalla. Cuando se entra el mandato, se coloca en *EXT como mensaje de petición.

3 Entonces el mandato se desplaza al final de la cola de mensajes de llamada QCMD y de aquí se pasa a QCMD.

4 Se analiza el mandato y se llama a su programa de proceso de mandatos (CPP).

5 El programa de proceso de mandatos envía mensajes de diagnóstico a la cola de mensajes de llamada para QCMD.

6 Entonces el programa de proceso de mandatos envía un mensaje de escape a la cola de mensajes de llamada para QCMD. El mensaje de escape notifica a QCMD que los mensajes de diagnóstico están en la cola y que QCMD debe finalizar el proceso de CPP.

7 QCMD supervisa la llegada de un mensaje de escape de comprobación de petición (CPF9901) o de comprobación de función (CPF9999). QCMD intenta entonces recibir el siguiente mensaje de petición. Si un procesador de peticiones recibe el mensaje CPF9901 ó CPF9999, debe ejecutar el mandato Reclamar Recursos (RCLRSC). El procesador de peticiones también debe supervisar los mensajes CPF1907 (finalizar petición) y CPF2415 (que indica que el usuario ha pulsado F3 o F12 en la pantalla de Entrada de Mandatos).

8 Puesto que se estaba procesando un mensaje de petición, todos los mensajes de la cola de mensajes de llamada para QCMD aparecen en la pantalla Entrada de Mandatos, que entonces solicita otro mandato al usuario de la estación de pantalla.

9 El mensaje de petición anterior (mandato) y sus mensajes asociados se graban en las anotaciones de trabajo según el nivel de registro de mensajes especificado para el trabajo. Para más información véase el apartado "Anotación de Mensajes" en el tema 8.7.

8.2.7.2 Escritura de Procedimientos y programas de Proceso de Peticiones

Especificar un procedimiento CL como un procesador de peticiones dentro de un programa ofrece numerosas ventajas. En la siguiente lista se especifican tres ventajas:

- Procesa los mensajes de petición tal como se describe en el apartado "Mensajes de Petición" en el tema 8.2.7.1.
- Permite utilizar el mandato Finalizar Petición (ENDRQS), que se puede emplear desde el menú de Petición del Sistema o como parte de la función de desconexión de trabajo.
- Permite filtrar los mensajes.

Para ser un procedimiento o programa de proceso de peticiones, su procedimiento o programa debe incluir los mandatos Enviar Mensaje de Programa (SNDPGMMSG) y Recibir Mensaje (RCVMSG). Por ejemplo, los siguientes mandatos permitirían a un procedimiento o programa convertirse en un procesador de peticiones:

```
SNDPGMMSG  MSG('Mensaje de petición') TOPGMQ(*EXT) MSGTYPE(*RQS)
RCVMSG     PGMQ(*EXT) MSGTYPE(*RQS) RMV(*NO)
```

El mensaje de petición se recibió de PGMQ *EXT. Cuando se recibe un mensaje de petición, éste se traslada a la cola de mensajes de llamada del procedimiento o programa especificado por el mandato RCVMSG. Por consiguiente, cuando se elimina el mensaje, debe utilizarse la cola de mensajes de llamada correcta.

Si el mensaje de petición se elimina mediante la clave de referencia de mensaje (MRK), la marca debe obtenerse de la palabra clave KEYVAR del mandato RCVMSG y no del mandato SNDPGMMSG. (La clave de referencia de mensaje cambia cuando el mensaje se desplaza al recibirse el mensaje. Debe especificarse RMV(*NO) en el mandato RCVMSG, ya que el procedimiento o programa no es un procesador de peticiones si el mensaje de petición se elimina de la cola de mensajes de llamada.

El procedimiento o programa se identifica como un procesador de peticiones cuando se recibe el mensaje de petición. Mientras el procedimiento o programa sea un procesador de peticiones, pueden finalizarse otros procedimientos o programas llamados utilizando la opción 2 (Finalizar petición) del menú Petición de Sistema. El procedimiento o programa procesador de peticiones debe incluir un supervisor para el mensaje CPF1907 (mandato MONMSG). Esto es necesario porque la función finalizar petición (desde la opción 2 del menú Petición de Sistema o el mandato Finalizar Petición) envía este mensaje al procesador de peticiones.

El procedimiento o programa permanece como procesador de peticiones hasta que finaliza el procedimiento (de forma normal o anómala) o hasta que se ejecuta un mandato RMVMSG para eliminar todos los mensajes de petición de la cola de mensajes de llamada del procesador de peticiones. Por ejemplo,el mandato siguiente elimina todos los mensajes de petición de la cola de mensajes y, por lo tanto, finaliza el proceso de peticiones:

```
RMVMSG CLEAR(*ALL)
```

8.2.7.3 Determinación de la existencia de un Procesador de Peticiones

Para determinar si un trabajo tiene un procesador de peticiones, visualice la pila de llamadas del trabajo. Utilice la opción 11 del mandato Visualizar Trabajo (DSPJOB) o Trabajar con Trabajo (WRKJOB), o seleccione la opción 10 para el trabajo listado en la pantalla WRKACTJOB. Si se muestra un número en la columna nivel de petición de la pantalla de la pila de llamadas del trabajo, el programa o procedimiento ILE asociado con el número es un procesador de peticiones. En el ejemplo siguiente, QCMD y QTEVIREF son procesadores de peticiones:

```
                            Visualizar Pila de Llamadas
                                                              Sistema:  S0000000
 Trabajo:  WS31           Usuario:  QSECOFR        Número:   000173

 Teclee opciones, pulse Intro
   5=Visualizar detalles

       Nivel    Programa o
 Opc  Petición  Procedimiento  Biblioteca     Sentencia      Instrucción
                QCMD           QSYS                             01DC
        1       QCMD           QSYS                             016B
                QTECADTR       QSYS                             0001
        2       QTEVIREF       QSYS                             02BA









                                                                      Final

 F3=Salir  F10=Actualizar pila   F11=Visualizar grupo activación  F12=Cancelar
 F17=Principio                   F18=Final
```

A continuación se presenta un ejemplo de un procedimiento de proceso de peticiones:

```
  PGM
      SNDPGMMSG  MSG('Mensaje de petición') TOPGMQ(*EXT) MSGTYPE(*RQS)
      RCVMSG     PGMQ(*EXT) MSGTYPE(*RQS) RMV(*NO)
        .
        .
        .
      CALL   PGM(PGMONE)
      MONMSG MSGID(CPF1907)
        .
        .
        .
      RMVMSG CLEAR(*ALL)
      CALL   PGM(PGMTWO)
        .
        .
        .
  ENDPGM
```

Los dos primeros mandatos del procedimiento lo convierten en un procesador de peticiones. El procedimiento permanece como procesador de peticiones hasta que se ejecuta el mandato RMVMSG. Se coloca un mandato de Supervisar Mensaje después de la llamada al programa PGMONE porque es posible que se envíe una petición de finalización desde PGMONE al procesador de peticiones. Si no se utiliza la supervisión, se producirá una comprobación de función para una petición de finalización. No se especifica ninguna supervisión de mensajes después de la llamada a PGMTWO porque el mandato RMVMSG finaliza el proceso de peticiones.

Si se intenta realizar una petición de finalización cuando se llama a un procedimiento o programa no de proceso de peticiones, se emite un mensaje de error y no se ejecuta la operación de finalización.

Nota: En los programas ejemplo, el mandato RCVMSG utiliza el número mínimo de parámetros que se necesitan para convertirse en procesador de peticiones. Es necesario indicar que desea recibir un mensaje de petición pero que no desea eliminarlo. También debe identificar la cola de llamadas específica desde la que se originó la petición de mensaje. Si es preciso, pueden añadirse otros parámetros.

8.2.8 Recuperación de Mensajes en un Procedimiento CL

Puede utilizar el mandato Recuperar Mensaje (RTVMSG) para recuperar el texto de un mensaje desde un archivo de mensajes a una variable. RTVMSG opera en descripciones de mensaje predefinidas. Puede especificar el identificador de mensaje y el nombre de archivo de mensajes además de lo siguiente:

- CCSID al que convertir. Especifica el identificador de juego de caracteres al que desea que se devuelvan el texto y los datos del mensaje.

- Campos de datos de mensajes. Los datos de mensajes para las variables de sustitución.

- CCSID de datos de mensaje. Especifica el identificador de juego de caracteres en el que se van a considerar los datos de mensaje suministrados.

- Un grupo de variables CL en las cuales se coloca la siguiente información (cada punto corresponde a una variable):

  - Mensaje (variable de tipo carácter, longitud variable)

  - Longitud del mensaje, incluida la longitud de los datos de la variable de sustitución (variable decimal, 5 posiciones decimales)

  - Información de ayuda de mensaje en línea (variable de tipo carácter, longitud variable)

  - Longitud de la ayuda del mensaje, incluida la longitud de los datos de las variables de sustitución (variable decimal, 5 posiciones decimales)

  - Código de gravedad (variable decimal, 2 posiciones decimales)

  - Opción de alerta (variable de tipo carácter, 9 caracteres)

  - Problema de anotación en las anotaciones de la actividad de servicio (variable de tipo carácter, 1 carácter)

  - El CCSID de los datos del mensaje es el identificador de juego de caracteres asociado a los datos de sustitución devueltos (variable decimal, 5 posiciones decimales)

  - El CCSID de los datos de texto es el identificador de juego de caracteres asociado al texto devuelto por los parámetros Mensaje y Ayuda de mensaje (variable decimal, 5 posiciones decimales)

Por ejemplo, el siguiente mandato añade la descripción del mensaje USR1001 al archivo de mensajes USRMSG:

```
ADDMSGD   MSGID(USR1001) MSGF(QGPL/USRMSG) +
          MSG('Archivo &1 no encontrado en biblioteca &2') +
          SECLVL('Cambiar nombre de archivo o de biblioteca') +
          SEV(40) FMT((*CHAR 10) (*CHAR 10))
```

Los siguientes mandatos dan como resultado la sustitución del nombre de archivo INVENT en la variable de 10 caracteres &FILE y del nombre de biblioteca QGPL en la variable &LIB. de 10 caracteres en el mensaje recuperado **USR1001.**

```
DCL &FILE TYPE(*CHAR) LEN(10) VALUE(INVENT)
DCL &LIB TYPE(*CHAR) LEN(10) VALUE(QGPL)
DCL &A TYPE(*CHAR) LEN(20)
DCL &MSG TYPE(*CHAR) LEN(50)
CHGVAR VAR(&A) VALUE(&FILE||&LIB)
RTVMSG   MSGID(USR1001) MSGF(QGPL/USRMSG) +
         MSGDTA(&A) MSG(&MSG)
```

Los datos de &1 y &2 se encuentran en la variable de procedimiento &A, en la que se han concatenado los valores de las variables de procedimiento &FILE y &LIB. El siguiente mensaje se coloca en la variable CL &MSG:

```
Archivo INVENT no encontrado en biblioteca QGPL
```

Si no se utiliza el parámetro MSGDTA en el mandato RTVMSG, el siguiente mensaje se coloca en la variable CL &MSG:

```
Archivo no encontrado en biblioteca
```

Después de que se coloca el mensaje en la variable &MSG, podría realizar

lo siguiente:

- Enviar el mensaje utilizando el mandato SNDPGMMSG
- Utilizar la variable como texto para una línea de mensajes en las DDS (M en la posición 38)
- Utilizar un subarchivo de mensajes
- Imprimir o visualizar el mensaje

Nota: No se puede recuperar el texto de un mensaje con los nombres de variable incluidos en el texto. RTVMSGD sirve para devolver un mensaje que se puede enviar. Para tener acceso al mensaje tal como se ha entrado en ADDMSGD, vea la herramienta CVTMSGF en QUSRTOOL.

8.2.9 Eliminación de Mensajes de una Cola de Mensajes

Los mensajes se retienen en una cola de mensajes hasta que se eliminan mediante el mandato Eliminar Mensaje (RMVMSG), el mandato Borrar Cola de Mensajes (CLRMSGQ), el parámetro RMV en los mandatos Recibir Mensaje (RCVMSG) y Enviar Respuesta (SNDRPY), las teclas de función de eliminación de la pantalla Visualizar Mensajes, o la opción de borrado de una cola de mensajes en la pantalla Trabajar con Cola de Mensajes. Le será posible eliminar:

- Un solo mensaje
- Todos los mensajes
- Todos los mensajes excepto los no respondidos
- Todos los mensajes antiguos
- Todos los mensajes nuevos
- Todos los mensajes de todos los programas inactivos

Para eliminar un solo mensaje utilizando el mandato RMVMSG o un solo mensaje antiguo mediante el mandato RCVMSG, especifique la clave de referencia del mensaje a eliminar.

Nota: La clave de referencia de mensaje también se puede utilizar para recibir y contestar a un mensaje.

Si elimina un mensaje de consulta al que no se ha respondido, se envía una respuesta por omisión al emisor del mensaje y se eliminan el mensaje de consulta y la respuesta por omisión. Si elimina un mensaje de consulta que ya ha respondido, se eliminan el mensaje y la respuesta.

Para eliminar todos los mensajes de todos los programas y procedimientos inactivos de una cola de mensajes de trabajo del usuario, especifique *ALLINACT para el parámetro PGMQ y *ALL para el parámetro CLEAR del mandato RMVMSG. Si desea imprimir sus anotaciones de trabajo antes de eliminar todos los mensajes inactivos, utilice el mandato Visualizar Anotaciones de Trabajo (DSPJOBLOG) y especifique *PRINT para el parámetro OUTPUT.

Al trabajar con una cola de mensajes de llamada de un procedimiento ILE, es posible que un mensaje de excepción para excepciones no manejadas se encuentre en la cola cuando se ejecute el mandato RMVMSG. Se puede utilizar la palabra clave RMVEXCP de este mandato para controlar las acciones para los mensajes de este tipo. Si se especifica *YES para esta palabra clave, el mandato RMVMSG provoca el manejo de la excepción y se elimina el mensaje. Si se especifica *NO, el mensaje no se elimina. Como resultado, la excepción no se maneja.

El siguiente mandato RMVMSG elimina un mensaje de la cola de mensajes de usuario denominada JONES. La clave de referencia de mensaje se encuentra en la variable CL &MRKEY.

```
DCL &MRKEY TYPE(*CHAR) LEN(4)
RCVMSG MSGQ(JONES) RMV(*NO) KEYVAR(&MRKEY)
RMVMSG MSGQ(JONES) MSGKEY(&MRKEY)
```

El siguiente mandato RMVMSG elimina todos los mensajes de una cola de mensajes.

```
RMVMSG CLEAR(*ALL)
```

Nota: El número máximo de mensajes en una cola de mensajes de usuario o una cola de mensajes de estación de trabajo es de 65.535 para cada tipo de mensaje enviado. Por ejemplo, en la cola puede haber 65.535 mensajes de diagnóstico; 65.535 mensajes de terminación, etc. En las colas de mensajes de llamada o *EXT, el número máximo de mensajes por tipo no está restringido.

8.3 Supervisión de Mensajes en un Programa o Procedimiento CL

Puede supervisar los mensajes de escape, de notificación o de estado que se envían a la cola de mensajes de llamada de su procedimiento o programa CL mediante los mandatos de su procedimiento o programa o mediante mandatos de otro procedimiento o programa. El mandato Supervisar Mensaje (MONMSG) supervisa los mensajes enviados a la cola de mensajes de llamada para ver si cumplen las condiciones especificadas en el mandato. Si la condición se cumple, se ejecuta el mandato CL especificado en el mandato MONMSG. La lógica que sigue el mandato MONMSG se indica a continuación:

Mensajes de Escape: Los mensajes de escape se envían para informar al procedimiento o programa de una condición de error que ha forzado al remitente a finalizar. Al supervisar los mensajes de escape, puede realizar acciones correctoras o efectuar limpieza, y finalizar su procedimiento o programa.

Mensajes de Estado o de Notificación: Los mensajes de estado y de notificación se envían para informar al procedimiento o programa de una condición anormal que no es lo bastante grave como para que el emisor finalice. Al supervisar si hay mensajes de estado o de notificación, el procedimiento o programa puede detectar esta condición y no permitir que la función continúe.

Puede supervisar los mensajes utilizando dos niveles de mandato MONMSG:

- Nivel de procedimiento: Puede supervisar los mensajes de escape, de notificación o de estado enviados por cualquier mandato en su procedimiento especificando el mandato MONMSG inmediatamente después del último mandato de declaración en su procedimiento o programa CL. Esto se llama mandato MONMSG a nivel de procedimiento. Puede utilizar 100 mandatos MONMSG a nivel de procedimiento en un procedimiento o programa OPM. (Un procedimiento CL puede contener como máximo 1000 mandatos MONMSG.) Esto permite manejar el mismo mensaje de escape de la misma forma para todos los mandatos. El parámetro EXEC es opcional y solamente puede tener especificado el mandato GOTO.

IMAGEN 9

- Nivel de mandato específico: puede supervisar un mensaje de escape, notificación o estado enviado por un mandato específico en su procedimiento o programa especificando el mandato MONMSG inmediatamente después del mandato. Esto se denomina mandato MONMSG a nivel de mandato. Para un único mandato se pueden utilizar hasta 100 mandatos MONMSG de nivel de mandato. Esto le permite manejar distintos mensajes de escape de varias maneras diferentes.

Para supervisar los mensajes de escape, estado o notificación, debe especificar en el mandato MONMSG los identificadores genéricos de los mensajes de una de las formas siguientes:

- **pppmmnn**

  Supervisa un mensaje específico. Por ejemplo, MCH1211 es el identificador del mensaje de escape de división por cero.

- **pppmm00**

  Supervisa cualquier mensaje con un identificador genérico de mensaje que empieza con un programa bajo licencia concreto (**ppp**) y los dígitos especificados por mm. Por ejemplo, CPF5100 indica que se supervisan todos los mensajes de notificación, estado y escape que empiezan por CPF51.

- **ppp0000**

  Supervisa cada mensaje que tenga un identificador genérico de mensaje que empiece con un programa bajo licencia concreto (**ppp**). Por ejemplo, **CPF0000** indica que se supervisan todos los mensajes de notificación, estado y escape que empiezan por **CPF**.

  Nota: No utilice **MONMSG CPF0000** al realizar una función del sistema, tal como instalar, salvar o restaurar todo el sistema, ya que puede perderse información importante.

- CPF9999

  Supervisa los mensajes de comprobación de función para todos los

identificadores genéricos de mensaje. Si no se supervisa un mensaje de error, se convierte en CPF9999 (comprobación de función).

Nota: Generalmente, al supervisar, el monitor también obtiene control cuando se envían mensajes de notificación y de estado.

Además de la supervisión de mensajes de escape mediante el identificador de mensaje, puede comparar una serie de caracteres, especificada en el mandato MONMSG, con datos enviados en el mensaje. Por ejemplo, el mandato siguiente supervisa un mensaje de escape (CPF5101) para el archivo MYFILE. El nombre del archivo se envía como datos de mensaje.

```
MONMSG MSGID(CPF5101) CMPDTA(MYFILE) EXEC(GOTO EOJ)
```

Los datos de comparación pueden tener una longitud máxima de 28 caracteres, y la comparación comienza con el primer carácter del primer campo de los datos de mensaje. Si los datos de comparación coinciden con los datos del mensaje, se ejecuta la acción especificada en el parámetro EXEC.

El parámetro EXEC del mandato MONMSG especifica cómo se ha de manejar un mensaje de escape. En este parámetro se puede especificar cualquier mandato excepto PGM, ENDPGM, IF, ELSE, DCL, DCLF, ENDDO y MONMSG. Puede especificar el mandato DO en el parámetro EXEC, en cuyo caso se ejecutan los mandatos del grupo DO. Cuando se ha ejecutado el mandato o grupo do (en el parámetro EXEC), el control vuelve al mandato de su procedimiento o mandato posterior al mandato que envió el mensaje de escape. Sin embargo, si especifica un mandato GOTO o RETURN, el control no se devuelve. Si no especifica el parámetro EXEC, el mensaje de escape se ignora y el procedimiento continúa.

A continuación se facilita un ejemplo de un mandato Cambiar Variable (CHGVAR) que se está supervisando para un mensaje de escape de división por cero, identificador de mensaje MCH1211:

```
CHGVAR VAR(&A) VALUE(&A/&B)
MONMSG MSGID(MCH1211) EXEC(CHGVAR VAR(&A) VALUE(1))
```

El valor de la variable &A se cambia por el valor de &A dividido por &B. Si &B es igual a 0, la operación de división no puede realizarse y se envía el mensaje de escape de división por cero al procedimiento. Cuando esto sucede, el valor de &A se cambia por 1 (tal como se ha especificado en el parámetro EXEC). También debe probar &B para cero y sólo realizar la división si no es cero.

En el siguiente ejemplo, el procedimiento supervisa el mensaje de escape CPF9801 (mensaje de objeto no encontrado) en el mandato Comprobar Objeto (CHKOBJ):

```
            PGM
            CHKOBJ LIB1/PGMA *PGM
            MONMSG MSGID(CPF9801) EXEC(GOTO NOTFOUND)
            CALL LIB1/PGMA
            RETURN
NOTFOUND:   CALL FIX001  /* PGMA Rutina No Encontrada */
            ENDPGM
```

El siguiente procedimiento CL contiene dos mandatos CALL y un mandato MONMSG a nivel de procedimiento para CPF0001. (Este mensaje de escape tiene lugar si un mandato CALL no se puede completar satisfactoriamente.) Si el mandato CALL falla, el procedimiento envía un mensaje de terminación y finaliza.

```
          PGM
          MONMSG MSGID(CPF0001) EXEC(GOTO ERROR)
          CALL PROGA
          CALL PROGB
          RETURN
ERROR:    SNDPGMMSG MSG('Un mandato CALL ha fallado') MSGTYPE(*COMP)
          ENDPGM
```

Si el parámetro EXEC no está codificado en un mandato MONMSG a nivel de procedimiento, se ignoran todos los mensajes de escape manejados por el mandato MONMSG. Si el mensaje de escape se produce en cualquier mandato excepto la condición de un mandato IF, el procedimiento o programa continúa procesando con el mandato que se habría ejecutado a continuación si no se hubiera producido el mensaje de escape. Si el mensaje de escape se produce en la condición de un mandato IF, el procedimiento o programa sigue procesando como si la condición del mandato IF fuera falsa. El siguiente ejemplo ilustra lo que sucede si se produce un mensaje de escape en diferentes puntos del procedimiento:

```
PGM
```

```
DCL &A TYPE(*DEC) LEN(5 0)
DCL &B TYPE(*DEC) LEN(5 0)
MONMSG MSGID(CPF0001 MCH1211)
CALL PGMA PARM(&A &B)
IF (&A/&B *EQ 5) THEN(CALL PGMB)
ELSE CALL PGMC
CALL PGMD
ENDPGM
```

Según dónde se produzca un mensaje de escape, sucede lo siguiente:

- Si CPF0001 se produce en la llamada a PGMA, el procedimiento reanuda el proceso en el mandato IF.
- Si MCH1211 (dividir por 0) se produce en el mandato IF, la condición IF se considera falsa, y el procedimiento reanuda el proceso con la llamada a PGMC.
- Si CPF0001 se produce en la llamada a PGMB o PGMC, el procedimiento reanuda el proceso con la llamada a PGMD.
- Si CPF0001 se produce en la llamada a PGMD, el procedimiento reanuda el proceso con el mandato ENDPGM, que provoca que se vuelva al procedimiento que efectúa la llamada.

También puede supervisar el mismo mensaje de escape para enviarlo mediante un mandato específico en su procedimiento o programa *y* mediante otro mandato. Ello requiere dos mandatos MONMSG. Uno de los mandatos MONMSG aparece a continuación del mandato que requiere un manejo especial para el mensaje de escape; para dicho mandato, el mandato MONMSG se utiliza cuando se envía el mensaje de escape. El otro mandato MONMSG sigue al último mandato de declaración para que se pueda utilizar este mandato MONMSG para todos los demás mandatos.

Los mandatos MONMSG sólo se aplican al procedimiento CL o programa OPM en que están codificados. Los mandatos MONMSG de un procedimiento no se aplican a otro procedimiento aunque los dos formen parte del mismo programa. El manual *Programación - Resumen de consulta* contiene una lista de los mensajes de escape, notificación y estado que se emiten para mandatos CL. También debe mantener una lista de todos los mensajes que ha definido.

Subtemas

8.3.1 Manejo por Omisión

Pueden enviarse muchos mensajes de escape a un procedimiento que llama a mandatos, programas y procedimientos. Probablemente no querrá supervisarlos ni manejarlos todos. Sin embargo, puede que desee supervisar y manejar los mensajes de escape que pertenecen a la función de su procedimiento. El sistema proporciona la supervisión y el manejo por omisión de los mensajes que no supervise.

El manejo por omisión da por sentado que se ha detectado un error en un procedimiento. Si está depurando el procedimiento, el mensaje se envía a su estación de pantalla. En este caso, puede entrar mandatos para analizar y corregir el error. Si no está depurando el procedimiento, el sistema ejecuta una función de filtraje de mensajes.

El filtraje de mensajes es una función de dos pasos que efectúa lo siguiente:

- Desplaza el mensaje de escape un paso antes en la pila de llamadas.
- Comprueba para ver si el procedimiento tiene un mandato MONMSG para el escape.

Si el procedimiento tiene un mandato MONMSG para el escape, la acción de filtraje de mensajes se detiene y se realiza la acción especificada por el mandato MONMSG. El filtraje de mensajes continúa hasta que se encuentra un mandato MONMSG o el límite de grupo de activación más cercano. Esto significa que el mensaje de escape no traspasa los límites del grupo de activación.

El proceso de comprobación de función empieza si se encuentra el límite del grupo de activación antes de que se encuentre un procedimiento con un mandato MONMSG que se aplica al mensaje. La acción en la excepción de escape original se considera completada. El mensaje de comprobación de función (CPF9999) se envía al procedimiento que era el destino del escape original. Si ese procedimiento tiene un mandato MONMSG para la comprobación de función, entonces se efectúa la acción especificada por ese mandato. Si no hay ningún mandato MONMSG especificado en ese procedimiento, se envía un mensaje de consulta al operador de la estación de trabajo si el trabajo es un trabajo interactivo. La operación de estación de trabajo puede responder con una de las siguientes respuestas;

R Reintentar el mandato que falla en el procedimiento.
I Ignorar el mensaje. Continuar el proceso en el siguiente mandato del procedimiento.
C Cancelar el procedimiento y dejar pasar la función de comprobación al procedimiento anterior de la pila de llamadas.
D Realizar un vuelco de la entrada de la pila de llamadas para el procedimiento que falla, cancelar el procedimiento y dejar pasar la función de comprobación al procedimiento anterior en la pila de llamadas. Esta es la acción por omisión si no se entra ninguna respuesta o si el trabajo es un trabajo de proceso por lotes.

Además, la comprobación de función no sale del límite del grupo de activación. Si la respuesta provoca que la comprobación de función salga de un límite de grupo de activación, se detienen las acciones posteriores en la comprobación de función. Todos los procedimientos hasta el límite del grupo de activación se cancelan y se envía el mensaje de escape CEE9901 a la entrada anterior de la pila de llamadas.

Puede supervisar los mensajes de escape de comprobación de función de forma que pueda:

- Realizar limpieza y finalizar el procedimiento
- Continuar con algún otro aspecto de su procedimiento

Nota: Si la descripción de mensaje para el escape no supervisado especifica una acción por omisión, se llama al programa de manejo por omisión antes de que se envíe el mensaje de comprobación de función. Cuando vuelve al programa de manejo por omisión, empieza el proceso de la comprobación de función.

8.3.2 Mensajes de Notificación

Además de supervisar los mensajes de escape, puede supervisar los mensajes de notificación que se envían a su cola de mensajes de llamada de su procedimiento o programa CL mediante los mandatos de su procedimiento o programa, o mediante los programas y procedimientos que llama. Los mensajes de notificación se envían para informar al procedimiento o programa de una condición que no es típicamente un error. Mediante la supervisión de los mensajes de notificación, puede especificar una acción distinta de la que especificaría si la condición no se hubiese detectado. Un número muy reducido de mandatos proporcionados por IBM envían mensajes de notificación.

La supervisión y el manejo de mensajes de notificación es parecido a la supervisión y manejo de mensajes de escape. La diferencia estriba en lo que sucede si no supervisa y maneja los mensajes de notificación. Los mensajes de notificación también se filtran de procedimiento a procedimiento dentro del límite del grupo de activación. Si se alcanza el límite del grupo de activación sin que se encuentre un mandato MONMSG para el mismo, se devuelve automáticamente la respuesta por omisión al remitente del mensaje de notificación y éste puede continuar procesando. A diferencia de los mensajes de escape, los mensajes de notificación no supervisados no se consideran una indicación de un error en el procedimiento o programa.

8.3.3 Mensajes de Estado

Puede supervisar mensajes de estado enviados por los mandatos del procedimiento CL o por los programas o procedimientos que llama. Los mensajes de estado indican al procedimiento el estado del trabajo efectuado por el remitente. Mediante la supervisión de los mensajes de estado, puede evitar que el programa o procedimiento siga con el proceso.

Para estos mensajes, no se almacena información de mensajes en una cola de mensajes. Por consiguiente, no puede recibirse un mensaje de estado.

Si no se supervisa un mensaje de estado, se salta igual que los mensajes de escape y de notificación. Si se alcanza el límite del grupo de activación sin que se encuentre un mandato MONMSG, la acción del mensaje se considera completa y el control se devuelve al emisor del mensaje para que siga procesando. Los mensajes de estado suelen enviarse para comunicar que se han detectado condiciones normales con las que el proceso puede continuar.

Los mensajes de estado enviados a una cola externa de mensajes se visualizan en la pantalla interactiva, e informan al usuario sobre una función que está en curso. Por ejemplo, el mandato Copiar Archivo (CPYF) envía un mensaje que informa al usuario de que se está realizando una operación de copia.

|Sólo se pueden enviar como mensajes de estado los mensajes predefinidos;
|los mensajes inmediatos no se pueden enviar. Puede utilizar el ID de mensaje proporcionado por el sistema, CPF9898, y facilitar datos de mensaje para enviar un mensaje de estado si no dispone de una descripción de mensaje ya existente.

Cuando la función ha terminado, el procedimiento o programa debe eliminar el mensaje de estado de la pantalla interactiva. El mensaje no puede eliminarse utilizando un mandato, pero si envía otro mensaje de estado a *EXT con un mensaje en blanco, parece que se elimina el mensaje. Para este fin puede utilizarse el ID CPI9801 proporcionado por el sistema. Cuando el control vuelve al programa OS/400, el mensaje *STATUS se puede borrar de la línea 24 sin enviar el mensaje CPI9801. El ejemplo siguiente muestra una aplicación frecuente de los ID de mensaje CPF9898 y CPI9801:

```
  SNDPGMMSG MSGID(CPF9898) MSGF(QCPFMSG) +
            MSGDTA('Función xxx en curso') +
            TOPGMQ(*EXT) MSGTYPE(*STATUS)
      □
      □   /*  Función de proceso  */
      □
  SNDPGMMSG MSGID(CPI9801) MSGF(QCPFMSG) +
            TOPGMQ(*EXT) MSGTYPE(*STATUS)
```

8.3.4 Impedir la Visualización de Mensajes de Estado

No puede evitar que los mandatos envíen mensajes de estado, pero sí puede evitar que los mensajes de estado se visualicen en la parte inferior de la pantalla.

Existen dos modos preferentes de evitar que aparezcan los mensajes de estado:

- Mandato Cambiar Perfil de Usuario (CHGUSRPRF)

  Puede cambiar el perfil de usuario de forma que siempre que se inicie la sesión utilizando dicho perfil, no se visualicen los mensajes de estado. Para ello, utilice el mandato CHGUSRPRF y especifique *NOSTSMSG en el parámetro Opción de Usuario (USROPT).

- Mandato Cambiar Trabajo (CHGJOB)

  Puede cambiar el trabajo que se está ejecutando actualmente de forma que los mensajes de estado no se visualicen. Para ello, utilice el mandato CHGJOB y especifique *NONE en el parámetro Mensaje de Estado (STSMSG). También puede utilizarse el mandato CHGJOB para ver los mensajes de estado especificando *NORMAL en el parámetro STSMSG.

Hay una tercera alternativa, menos utilizada, que consiste en utilizar el mandato Alterar Temporalmente Archivo de Mensajes (OVRMSGF) y sustituir los identificadores de mensaje de estado por un mensaje en blanco.

8.4 Programas Manejadores de Interrupciones

Un programa manejador de interrupciones es aquél al que se llama automáticamente cuando un mensaje llega a una cola de mensajes que se encuentra en modalidad *BREAK. El nombre del programa y el nombre de la entrega con interrupción deben estar especificados en el mismo mandato Cambiar Cola de Mensajes (CHGMSGQ). Aunque el programa está especificado en el mandato CHGMSGQ, son uno o más procedimientos dentro del programa los que procesan el mensaje. Un procedimiento dentro de este programa debe ejecutar un mandato Recibir Mensaje (RCVMSG) para recibir el mensaje. Para recibir y manejar el mensaje, se pasan parámetros al programa definido por el usuario llamado para manejar mensajes para entrega con interrupción (más específicamente, estos parámetros se pasan al primer procedimiento a ejecutarse dentro del programa). Los parámetros identifican la cola de mensajes y a la clave de referencia de mensaje (MRK) del mensaje causante de la interrupción. Consulte el programa de salida de Manejador de Interrupciones en el manual *System API Reference*, SC41-4801, para obtener una lista de parámetros. Si se llama al programa manejador de interrupciones, éste interrumpe el trabajo en el que aparece el mensaje y se ejecuta. Cuando finaliza el programa manejador de interrupciones, el programa original reanuda el proceso.

El siguiente programa (PGMA), que consta de sólo este procedimiento, es un ejemplo de un programa de manejo de interrupciones.

```
  PGM PARM(&MSGQ &MSGLIB &MRK)
  DCL VAR(&MSGQ) TYPE(*CHAR) LEN(10)
  DCL VAR(&MSGLIB) TYPE(*CHAR) LEN(10)
  DCL VAR(&MRK) TYPE(*CHAR) LEN(4)
  DCL VAR(&MSG) TYPE(*CHAR) LEN(75)
  RCVMSG  MSGQ(&MSGLIB/&MSGQ) MSGKEY(&MRK) +
          MSG(&MSG)
  .
  .
  .
  ENDPGM
```

Después de la creación del programa manejador de interrupciones, la ejecución del siguiente mandato lo conecta con la cola de mensajes QSYSMSG.

```
  CHGMSGQ   MSGQ(QSYS/QSYSMSG) DLVRY(*BREAK) PGM(PGMA)
```

Notas:

1. Cuando se manejan los mensajes, éstos deben eliminarse de la cola de mensajes. Cuando una cola de mensajes entra en la modalidad de interrupción, cualquier mensaje contenido en ella hará que se llame al programa manejador de interrupciones.

2. El procedimiento o programa que recibe el mensaje no debe estar codificado con un tiempo de espera distinto de cero para recibir el mensaje. Especifique un valor que no sea cero para el parámetro de espera con el mandato Recibir Mensaje (RCVMSG). El sistema no puede manejar la llegada del mensaje mientras el trabajo está ejecutando un evento de manejo de interrupciones.

Un ejemplo de utilización de un programa manejador de interrupciones es que éste envíe un mensaje, que normalmente se envía a la cola QSYSOPR, a otra cola en lugar de QSYSOPR o además de a ésta.

El siguiente ejemplo ilustra un programa definido por el usuario (de nuevo con un solo procedimiento) para manejar mensajes de interrupción. Cuando se utiliza este programa, el usuario de la estación de pantalla no necesita responder a los mensajes CPA5243 (Pulsar Listo, Inicio o Parar-Iniciar en el dispositivo &1) y CPA5316 (Verificar alineación en dispositivo &3).

```
  BRKPGM:     PGM (&MSGQ &MSGQLIB &MSGMRK)
              DCL &MSGQ TYPE(*CHAR) LEN(10)
              DCL &MSGQLIB TYPE(*CHAR) LEN(10)
              DCL &MSGMRK TYPE(*CHAR) LEN(4)
              DCL &MSGID TYPE(*CHAR) LEN(7)
              RCVMSG MSGQ(&MSGQLIB/&MSGQ) MSGKEY(&MSGMRK) +
                     MSGID(&MSGID) RMV(*NO)
              /* Pasar por alto mensaje CPA5243 */
              IF (&MSGID *EQ 'CPA5243') GOTO ENDBRKPGM
              /* Responder a mensaje de alineación de formularios */
              IF (&MSGID *EQ 'CPA5316') +
                     DO
                     SNDRPY MSGKEY(&MSGMRK) MSGQ(&MSGQLIB/&MSGQ) RPY(I)
                     ENDDO
              /* Otros mensajes requieren la intervención del usuario */
```

```
             ELSE CMD(DSPMSG MSGQ(&MSGQLIB/&MSGQ))
  ENDBRKPGM:  ENDPGM
```

Aviso: En el ejemplo anterior de programa manejador de interrupciones, si un mensaje CPA5316 llegara a la cola mientras se ejecuta el mandato DSPMSG, la pantalla DSPMSG muestra el mensaje original que provocó la interrupción y el mensaje CPA5316. La pantalla DSPMSG espera a que el operador responda a dicho mensaje antes de continuar.

Nota: Este programa no puede abrir un archivo de pantalla si el programa interrumpido espera los datos de entrada de la pantalla.

Puede utilizar la lista de respuestas del sistema para indicar que éste emitirá una respuesta a los mensajes de consulta predefinidos. Por esta razón, el usuario de la estación de pantalla no necesita responder. Para obtener más información, véase el apartado "Utilización de la Lista de Respuestas del Sistema" en el tema 8.6.

Un procedimiento dentro de un programa de usuario de manejo de interrupciones, puede necesitar un procedimiento Suspender o Restaurar para garantizar que la pantalla se suspende y restaura mientras se está ejecutando la función de manejo de mensajes. El procedimiento de Suspender y Restaurar es necesario sólo si existe la siguiente condición:

- Un procedimiento en el programa de interrupción visualiza otros menús o pantallas
- El programa de interrupción llama a otros programas que pueden visualizar otros menús o pantallas.

El siguiente ejemplo clarifica el procedimiento y el archivo de pantalla de usuario necesario para suspender y restaurar la pantalla:

Nota: Debe especificar RSTDSP(*YES) para crear el archivo de pantalla.

```
          A          R SAVFMT                      OVERLAY  KEEP
          A*
          A          R DUMMY                       OVERLAY
          A                                        KEEP
          A                                        ASSUME
          A            DUMMYR         1A     1  2DSPATR(ND)


     PGM PARM(&MSGQ &MSGLIB &MRK)
         DCL VAR(&MSGQ) TYPE(*CHAR) LEN(10)
         DCL VAR(&MSGLIB) TYPE(*CHAR) LEN(10)
         DCL VAR(&MRK) TYPE(*DEC) LEN(4)
         DCLF FILE(UDDS/BRKPGMFM)
         SNDF RCDFMT(SAVFMT)
         CALL PGM(Programa Interrupción Usuario)
         SNDF RCDFMT(SAVFMT)
     ENDPGM
```

Si no desea que el programa de usuario de manejo de interrupciones interrumpa el trabajo interactivo, el programa puede someterse para que se ejecute por lotes. Para ello, especifique un programa de manejo de interrupciones que reciba el mensaje y a continuación ejecute un SBMJOB. SBMJOB ejecuta una llamada al programa de manejo de interrupciones actual con cualquier parámetro que desee utilizar. (Por ejemplo, información de recibir mensaje.) Entonces el control se devuelve al trabajo interactivo y continuará normalmente.

Puede encontrar otro ejemplo de programa manejador de interrupciones en QUSRTOOL con la herramienta STSMSG, que le permite convertir parte o la totalidad de los mensajes recibidos en la estación de trabajo en mensajes de estado.

8.5 Cola de Mensajes QSYSMSG

QSYSMSG es una cola de mensajes opcional que puede crearse en la biblioteca QSYS. Si existe y no está dañada, ciertos mensajes se dirigen a ella en lugar de a la cola de mensajes QSYSOPR, o además de a ésta. Ello permite que un programa escrito por el usuario obtenga el control cuando se envían determinados mensajes. No debe crear la cola QSYSMSG a menos que desee que reciba mensajes específicos.

Para crear la cola QSYSMSG, entre el mandato siguiente:

```
  CRTMSGQ  QSYS/QSYSMSG +
           TEXT('MSGQ opcional para recibir mensajes específicos del sistema')
```

Una vez que se ha creado la cola de mensajes QSYSMSG, todos los mensajes específicos (que aparecen más adelante en el apartado "Mensajes Enviados a la Cola de Mensajes QSYSMSG" en el tema 8.5.1) se dirigen a ella. Puede escribir un programa para recibir mensajes para los cuales puede realizar una acción especial y enviar otros mensajes a la cola de mensajes QSYSOPR o a otra cola de mensajes. Este programa debe escribirse como manejador de interrupciones.

Subtemas

8.5.1 Mensajes Enviados a la Cola de Mensajes QSYSMSG

A continuación se describen los mensajes específicos enviados a la cola QSYSMSG:

CPF0907 Puede existir una condición grave de almacenamiento. Pulse Ayuda.

Este mensaje se envía si la cantidad de almacenamiento auxiliar disponible en la agrupación de almacenamiento auxiliar del sistema ha alcanzado el valor de umbral.

Puede utilizarse la función de las herramientas de servicio del sistema para visualizar u modificar el valor de umbral. Para obtener más información consulte el manual Backup and Recovery - Advanced.

CPF111C Sistema planificado para apagarse.

El sistema ha detectado una manipulación con la información sobre licencia OS/400. El sistema se apagará automáticamente dentro de dos horas si no restaura la información sobre licencia.

Utilice los siguientes mandatos para restaurar la información sobre licencia del producto:

- Utilice el mandato Trabajar con Objeto para suprimir la información sobre el producto:

  ```
  WRKOBJ QSYS/QSZ0050 *PRDDFN
  ```

- Utilice el mandato Restaurar Objeto para restaurar la información sobre el producto:

  ```
  RSTOBJ OBJ(QSZ0050) DEV(nombre dispos.) SAVLIB(QSYS) OPTION(*NEW)
  ```

CPF111D Se apaga el sistema.

El sistema ha detectado una manipulación con la información sobre licencia de producto OS/400. El sistema se está apagando.

El sistema se apagará. Utilice los siguientes mandatos para impedir que el sistema se apague dos veces:

- Utilice el mandato Trabajar con Objeto para suprimir la información sobre el producto:

  ```
  WRKOBJ QSYS/QSZ0050 *PRDDFN
  ```

- Utilice el mandato Restaurar Objeto para restaurar la información sobre el producto:

  ```
  RSTOBJ OBJ(QSZ0050) DEV(nombre dispos.) SAVLIB(QSYS) OPTION(*NEW)
  ```

CPF1269 Se ha rechazado una petición de arranque de programa recibida en el dispositivo de comunicaciones con códigos de razón.

Este mensaje se envía cuando se ha rechazado una petición de arranque; contiene un código de razón que indica por qué se produjo el rechazo. Para ver una explicación detallada de cada código de razón, consulte la descripción de mensaje para CPF1269 en el manual ICF Programming.

Si una contraseña no es válida o se produce una condición de falta de autorización al utilizar APPC, puede indicar que un trabajo normal contiene errores o que alguien está intentando violar la seguridad. Puede evitar que se siga utilizando la descripción de dispositivo APPC hasta que se comprenda la condición efectuando lo siguiente:

- Enviar el mensaje a la cola de mensajes QSYSOPR.
- Grabar el intento para que el responsable de seguridad lo revise.
- Emitir el mandato Finalizar Modalidad (ENDMOD) para establecer los trabajos permitidos a cero. Esto permite que los trabajos que utilizan actualmente la descripción de dispositivo "peer" permanezcan activos, pero impide que otros trabajos arranquen hasta que no se comprenda la condición.

- Contar el número de intentos en un período de tiempo dado. Puede establecerse un umbral en el programa para el número de intentos que no han sido válidos antes de emprender una acción de más importancia (como cambiar el número máximo de sesiones a cero). Puede asignar este valor de umbral por unidad de identificador de trabajo (que puede ser blanco), por descripción de dispositivo APPC o para todo el entorno APPC.

CPF1393 El subsistema &1 inhabilitó el perfil de usuario &2 en el dispositivo &3.

Este mensaje se envía cuando un usuario ha intentado conectarse varias veces, lo que ha provocado que se inhabilite el perfil de usuario.

CPF1397 El subsistema ha desactivado la estación de trabajo.

Este mensaje se envía si se alcanza el valor de umbral asignado por valor del sistema QMAXSIGN y se desactiva el dispositivo. El mensaje indica que un usuario no está entrando una contraseña válida. Los datos del mensaje de CPF1397 contiene el nombre del dispositivo desde el que se envió el mensaje. Puede utilizar esta información y diseñar un programa para efectuar la acción adecuada. Puede considerar la posibilidad de realizar una o varias de las acciones siguientes:

- Enviar el mismo mensaje a la cola de mensajes QSYSOPR
- Grabar el intento para que el responsable de seguridad lo revise.
- Activar automáticamente el dispositivo después de un retardo de tiempo significativo

CPI0948 Protección por duplicación de disco suspendida en la unidad de disco &1.

El sistema no ha podido localizar una unidad de almacenamiento. No se han perdido datos.

CPI0949 Protección por duplicación de disco suspendida en la unidad de disco &1.

La protección por duplicación de disco queda anulada.

CPI0950 Unidad de almacenamiento disponible ahora.

Queda disponible en este momento una unidad de almacenamiento que no estaba en la configuración. No se han perdido datos.

CPI0953 Se ha alcanzado el umbral de almacenamiento de la ASP.

Este mensaje se envía si la cantidad de almacenamiento disponible en la agrupación de almacenamiento auxiliar (ASP) especificada ha alcanzado el valor de umbral. Los datos del mensaje CPI0953 contienen la capacidad de almacenamiento auxiliar, el almacenamiento auxiliar utilizado, el porcentaje de umbral y el porcentaje de almacenamiento auxiliar disponible. Puede utilizar esta información para llevar a cabo la acción adecuada.

CPI0954 Se ha excedido el límite de almacenamiento en la ASP.

Este mensaje se envía si se ha utilizado todo el almacenamiento disponible en la ASP especificada.

CPI0955 Se ha excedido el límite del almacenamiento no protegido de la ASP del sistema.

Este mensaje se envía si se ha utilizado todo el almacenamiento disponible en la ASP del sistema.

CPI0964 Existe una condición de batería agotándose.

Este mensaje se envía si la fuente de alimentación ininterrumpible externa o la batería interna indican una condición de batería agotándose.

CPI0965 Anomalía en el dispositivo de reserva de batería de la unidad del sistema.

Este mensaje se envía si se produce una anomalía en la batería o en el cargador de la misma en el dispositivo de unidad de alimentación de la batería de la unidad del sistema.

CPI0966 Anomalía en el dispositivo de reserva de batería de la unidad de expansión.

Este mensaje se envía si se produce una anomalía en la batería o en el cargador de la misma en el dispositivo de unidad de alimentación de la batería de la unidad de expansión.

CPI0988 La protección por duplicación de disco se está reanudando en la unidad de disco &1.

Este mensaje se envía si la sincronización de duplicación de disco de una unidad de disco se ha iniciado y la protección por duplicación de disco se reanuda. Uno de los pasos que realiza el sistema antes de que se reanude la protección por duplicación de disco es la copia de datos de una unidad de disco a otra para que los datos de ambas unidades de disco sean los mismos. Puede observarse una disminución del rendimiento del sistema mientras se copian los datos. Una vez finalizada la copia de los datos del disco, se envía el mensaje CPI0989 a esta cola de mensajes y se reanuda la protección por duplicación de disco.

CPI0989 Protección por duplicación de disco reanudada en la unidad de disco &1.

Este mensaje se envía si la sincronización de la duplicación de disco de una unidad de disco ha finalizado satisfactoriamente. El sistema ha terminado la copia de datos de una unidad de disco a la otra. Se reanuda la protección por duplicación de disco.

|**CPI0998 Se ha producido un error en la unidad de disco &1.**

| Este mensaje se envía si se detectaron errores en la unidad de
| disco &1. El mensaje no contiene información sobre la anomalía
| para ejecutar el análisis de problemas.

|**CPI1117 Se ha suprimido la planificación de trabajos &1 dañada en
| biblioteca &2**

| Este mensaje se envía cuando se suprime la planificación de
| trabajos en la biblioteca debido a un daño.

|**CPI1136 La protección por duplicidad de disco sigue suspendida.**

| Este mensaje se envía cada hora si la protección por duplicación
| de disco todavía se encuentra suspendida en una o más unidades
| de disco.

CPI1138 Se ha recuperado el desbordamiento de almacenamiento.

Este mensaje se envía cuando la ASP &1 ya no tiene objetos que se han desbordado en la ASP del sistema por la razón &2.

CPI1139 Error en recuperación de desbordamiento de almacenamiento.

Este mensaje se envía cuando falla un intento de recuperarse del desbordamiento de almacenamiento.

CPI1153 Ha finalizado el período de ignorar contraseña del sistema.

Este mensaje se envía cuando el sistema ha estado trabajando mientras estaba en vigor el período de ignorar contraseña del sistema. Dicho período ha finalizado. A menos que se facilite la contraseña del sistema correcta, la siguiente IPL no podrá completarse satisfactoriamente.

CPI1154 El período de ignorar contraseña del sistema finalizará dentro de &5 días.

Este mensaje se envía cuando la contraseña del sistema no se ha entrado o no se ha entrado correctamente (durante una IPL anterior) y estaba seleccionado el período de ignorar del sistema.

|**CPI1159 ID de sistema caducará si se llevan a cabo &1 instalaciones más.**

| Este mensaje se envía cuando el ID de sistema esta a punto de
| caducar. Debe contactarse con el servicio técnico IBM.

|**CPI1160 Ha caducado el ID de sistema.**

| Este mensaje se envía cuando el identificador de sistema ha
| caducado. Debe contactarse con el servicio técnico IBM

|**CPI1161 Unidad &1 con protección de paridad de dispositivo, no**

| **totalmente operativa**

| La unidad &1 es un componente de un subsistema de unidades de
| discos con protección por paridad de dispositivos. La unidad &1
| necesita servicio. Los datos se han salvado. Si no se corrige
| esta condición, se pueden producir una reducción de rendimiento,
| errores de máquina, y posibles pérdidas de datos.

|**CPI1162** **Unidad &1 con protección de paridad de dispositivo, no**
| **totalmente operativa**

| La unidad &1 es un componente de un subsistema de unidades de
| discos con protección por paridad de dispositivos. La unidad &1
| no está totalmente operativa por una de las razones siguientes:

| - El representante del servicio técnico está reparando la
| unidad.
| - La unidad no está operativa, pero no hay suficiente
| información para ejecutar el análisis del problema.

|**CPI1165** **Una o más unidades de protección de paridad de dispositivo**
| **todavía no están completamente operativas.**

| Una o más unidades de los subsistemas de unidades de discos con
| protección por paridad de dispositivos, no están todavía
| completamente operativas debido a errores.

|**CPI1166** **Unidades con protección de paridad por dispositivo totalmente**
| **operativas.**

| Las unidades para todos los subsistemas IOP que proporcionan
| protección por paridad de dispositivos están completamente
| operativas.

|**CPI1167** **Se ha producido un error de procesador de E/S temporal.**

| Se ha producido una condición de error en un procesador de E/S
| con dispositivos de disco.

|**CPI1168** **Se ha producido un error en la unidad de disco &1.**

| Unidad de discos &1 encontró un error. Se ha podido dañar un
| objeto. Si se agrava el problema, puede producirse un error de
| máquina. Lo siguiente identifica la unidad de discos.

|**CPI1169** **Unidad de disco &1 no operativa.**

| La unidad de discos &1 ha dejado de operar. No se han perdido
| datos.

CPI1393 **Subsistema &1 inhabilitó perfil de usuario &2 en dispositivo &3.**

Este mensaje se envía cuando el usuario alcanza el número máximo de intentos de conexión determinado por el valor del sistema QMAXSIGN. La acción que se lleva a cabo cuando se alcanza este valor viene determinada por el valor del sistema QMAXSGNACN.

CPI2209 **Perfil de usuario &1 suprimido debido a que fue dañado.**

Este mensaje se envía cuando se suprime un perfil de usuario porque estaba dañado. Es posible que dicho perfil poseyera objetos antes de su supresión. Ahora, estos objetos no tienen propietario. Puede utilizarse un mandato Reclamar Almacenamiento (RCLSTG) para transferir la propiedad de estos objetos al perfil de usuario QDFTOWN.

CPI2283 **El valor del sistema QAUDCTL ha cambiado a *NONE.**

Este mensaje se envía cada hora después de que el sistema desactive el seguimiento debido a que éste ha fallado. Para activar de nuevo el seguimiento o para determinar por qué ha fallado, puede cambiar el valor del sistema QAUDCTL por un valor que no sea *NONE.

CPI2284 **El valor del sistema QAUDCTL ha cambiado a *NONE.**

Este mensaje se envía durante la IPL si el sistema ha desactivado el seguimiento porque éste ha fallado. Para activar de nuevo el seguimiento o para determinar por qué ha fallado, puede cambiar el valor del sistema QAUDCTL por un valor que no sea *NONE.

CPI8A13 **La biblioteca QDOC está cerca del límite de objetos del sistema.**

Este mensaje se envía cuando la cantidad de objetos situados en la biblioteca QDOC se acerca al número máximo de objetos que el sistema puede almacenar en una sola biblioteca.

CPI8A14 La biblioteca QDOC ha excedido el límite de objetos del sistema.

Este mensaje se envía cuando la cantidad de objetos situados en la biblioteca QDOC ha sobrepasado el número máximo de objetos que el sistema soporta en una biblioteca.

CPI8898 Se ha detectado pérdida de señal óptica en bus óptico.

Este mensaje se envía cuando se detecta una anomalía en un bus óptico. El bus funciona en modalidad reducida. Este mensaje se anotará en las anotaciones de actividades de servicio y tiene disponible la opción PAR.

CPI9014 Se ha recibido del dispositivo una contraseña no válida.

Este mensaje se envía cuando se ha recibido una contraseña incorrecta en una sesión de intercambio de documento. Puede indicar intentos no autorizados para acceder al sistema.

CPI9490 Error de disco en el dispositivo &25.

Este mensaje se envía cuando se ha detectado un error de disco.

CPI94A0 Error de disco en el dispositivo &25.

Este mensaje se envía cuando se ha detectado un error de disco.

CPI94CE Error detectado en cables, Procesador del Sistema, adaptador de extensión del bus o adaptador de expansión del bus.

Este mensaje se envía cuando el sistema detecta una anomalía en el almacenamiento principal. El rendimiento del sistema puede verse reducido. Ejecute el análisis de problemas para determinar cuál es la tarjeta que presentan la anomalía.

CPI94CF Se ha detectado un error de tarjeta de Almacenamiento Principal.

Este mensaje se envía cuando el sistema detecta una anomalía en el almacenamiento principal. El rendimiento del sistema puede verse reducido. Ejecute el análisis de problemas para determinar cuál es la tarjeta que presentan la anomalía.

CPI94FC Error de disco en el dispositivo &25.

Este mensaje se envía cuando la Unidad de Disco 9336 ha determinado que uno de sus componentes excede el umbral de error y comienza a presentar anomalías.

CPP0DD9 Se ha detectado un error de procesador del sistema.

Este mensaje se envía cuando se produce una anomalía en un procesador del sistema o en la antememoria de éste. El rendimiento del sistema puede verse reducido.

CPP0DDA Se ha detectado una anomalía de un procesador del sistema en la ranura 9.

Este mensaje se envía cuando se produce una anomalía en un procesador del sistema o en la antememoria de éste. El rendimiento del sistema puede verse reducido.

CPP0DDB Se ha detectado una anomalía de procesador del sistema en la ranura 10.

Este mensaje se envía cuando se produce una anomalía en un procesador del sistema o en la antememoria de éste. El rendimiento del sistema puede verse reducido.

CPP0DDC Se ha detectado un error de procesador del sistema.

Este mensaje se envía cuando el sistema detecta un error en el procesador del sistema. El rendimiento del sistema puede verse reducido.

CPP0DDD Los diagnósticos del procesador del sistema han detectado un error.

Este mensaje se envía cuando los diagnósticos del procesador del sistema detectan una anomalía durante la IPL, pero el sistema puede seguir funcionando.

CPP0DDE Se ha detectado un error en un procesador del sistema.

Este mensaje se envía cuando se detecta una anomalía de control en un procesador del sistema. La comprobación y corrección del hardware soluciona la anomalía. Sin embargo, si ha efectuado una carga del programa inicial (IPL), no puede inicializarse el control y el sistema se reconfigurará sin dicho procesador.

CPP0DDF Falta uno de los procesadores del sistema.

Este mensaje se envía cuando falta un procesador en un sistema de multiprocesador.

CPP29B0 Se ha excedido el umbral de recuperación en el dispositivo &25.

Este mensaje se envía cuando uno de los componentes de la unidad de disco 9337 comienza a funcionar incorrectamente.

CPP29B8 Se ha suspendido la protección por paridad de dispositivos en el dispositivo &25.

Este mensaje se envía cuando uno de los componentes de la batería de discos 9337 comienza a fallar. La protección para la implantación de la técnica RAID 5 se ha suspendido en la batería de discos.

CPP29B9 Se ha suspendido la protección de alimentación en el dispositivo &25.

Este mensaje se envía cuando uno de los módulos de alimentación de la batería de discos 9337 funciona incorrectamente. La protección de alimentación se ha suspendido en la batería de discos.

CPP29BA Error de hardware en el dispositivo &25.

Este mensaje se envía cuando uno de los componentes de la batería de discos 9337 ha fallado. Es preciso dar servicio técnico.

CPP951B Error en la unidad de alimentación por batería.

Este mensaje se envía cuando ha fallado la unidad de alimentación de batería.

CPP9522 Error en la unidad de alimentación por batería.

Este mensaje se envía cuando ha fallado la unidad de alimentación por batería en la Unidad de Expansión 5042 o en la Unidad de Extensión 5040.

CPP955E No está instalada una unidad de alimentación por batería.

Este mensaje se envía cuando en la fuente de alimentación de la Unidad del Sistema 9406 no hay instalada una unidad de alimentación por batería.

CPP9575 La unidad de alimentación por batería en la 9406 necesita sustituirse.

Este mensaje se envía cuando la unidad de alimentación por batería de la Unidad del Sistema 9406 ha presentado anomalías y ha de sustituirse. Es posible que pueda seguir funcionando, pero se han producido más casos de ciclos de carga y descarga de los recomendados.

CPP9576 La unidad de alimentación por batería en la 9406 necesita sustituirse.

Este mensaje se envía cuando la unidad de alimentación por batería de la Unidad del Sistema 9406 ha presentado anomalías y ha de sustituirse. Es posible que pueda seguir funcionando, pero ha estado instalada más tiempo del recomendado.

CPP9589 La prueba de la unidad de alimentación por batería se ha completado.

Este mensaje se envía cuando se ha completado la prueba de la unidad de alimentación por batería y los resultados se han anotado.

CPP9616 No está instalada una unidad de alimentación por batería.

Este mensaje se envía cuando la unidad de alimentación por batería no está instalada en la fuente de alimentación de la Unidad de Expansión 5042 o de la Unidad de Extensión 5040.

CPP9617 Es necesario sustituir una unidad de alimentación por batería.

Este mensaje se envía cuando es preciso cambiar la unidad de alimentación por batería en la Unidad de Expansión 5042 o en la Unidad de Extensión 5040. Es posible que pueda seguir funcionando, pero se han producido más casos de ciclos de carga y descarga de los recomendados.

CPP9618 Es necesario sustituir una unidad de alimentación por batería.

Este mensaje se envía cuando es preciso cambiar la unidad de alimentación por batería en la Unidad de Expansión 5042 o en la Unidad de Extensión 5040. Es posible que pueda seguir funcionando, pero ha estado instalada más tiempo del recomendado.

CPP961F Error en Módulo General CC 3.

Este mensaje se envía cuando el módulo general cc 3 de la Unidad del Sistema 9406 ha presentado anomalías.

CPP9620 Error en Módulo General CC 2.

Este mensaje se envía cuando el módulo general cc 2 de la Unidad del Sistema 9406 ha presentado anomalías.

CPP9621 Error en Módulo General CC 1.

Este mensaje se envía cuando el módulo general cc 1 de la Unidad del Sistema 9406 ha presentado anomalías.

CPP9622 Error en Módulo General CC 1.

Este mensaje se envía cuando el módulo general cc 1 de la Unidad de Expansión 5042 o de la Unidad de Extensión 5040 ha presentado alguna anomalía. Otros módulos generales cc pueden producir también el mismo error.

CPP9623 Error en Módulo General CC 2.

Este mensaje se envía cuando el módulo general cc 2 de la Unidad de Expansión 5042 o de la Unidad de Extensión 5040 ha presentado alguna anomalía. Otros módulos generales cc también pueden producir este error.

CPP962B Error en Módulo General CC 3.

Este mensaje se envía cuando el módulo general cc 3 de la Unidad de Expansión 5042 o de la Unidad de Extensión 5040 ha presentado alguna anomalía. Otros módulos generales cc pueden producir también el mismo error.

8.5.2 Ejemplo de Programa para Recibir Mensajes de QSYSMSG

A continuación se facilita un programa ejemplo que recibe mensajes de la cola de mensajes QSYSMSG. El programa consta de un solo procedimiento que recibe mensajes y maneja el mensaje CPF1269. El código de razón del mensaje CPF1269 está en formato binario. Debe convertirse en un valor decimal para las comparaciones con los códigos de razón 704 y 705. El procedimiento emite el mandato ENDMOD para evitar que se arranquen nuevos trabajos hasta que se comprenda cuál es la situación. Después envía el mismo mensaje a una cola de mensajes definida por el usuario para que el responsable de seguridad lo revise. Asimismo, envía un mensaje al operador del sistema para informarle de lo que ha ocurrido. Si se recibe un mensaje distinto, se envía al operador del sistema.

Para llamar a este programa ejemplo, se arranca un trabajo por separado. El trabajo permanece activo, esperando la llegada de un mensaje. El trabajo puede finalizarse utilizando el mandato ENDJOB.

```
   /*****************************************************************/
   /*                                                               */
   /*  Ejemplo de programa para recibir mensajes de QSYSMSG         */
   /*                                                               */
   /*****************************************************************/
   /*                                                               */
   /*  El programa busca el mensaje CPF1269 con código de razón 704 */
   /*  ó 705. Si lo encuentra, notifica a QSECOFR el error de       */
   /*  seguridad. Si no, vuelve a enviar el mensaje a QSYSOPR.      */
   /*                                                               */
   /*  A continuación se describe el mensaje CPF1269                */
   /*                                                               */
   /*      CPF1269:  Petición arranque programa recibida en dispositivo*/
   /*                &1 fue rechazado con códigos de razón &6, &7   */
   /*                                                               */
   /*      Datos mensaje de DSPMSGD CPF1269                         */
   /*                                                               */
   /*        Datos tipo   despl   long  Descripción                 */
   /*                                                               */
   /*         &1   *CHAR      1     10  Dispositivo                 */
   /*         &2   *CHAR     11      8  Modo                       */
   /*         &3   *CHAR     19     10  Trabajo - número            */
   /*         &4   *CHAR     29     10  Trabajo - usuario           */
   /*         &5   *CHAR     39      6  Trabajo - nombre            */
   /*         &6   *BIN      45      2  Código de razón - mayor     */
   /*         &7   *BIN      47      2  Código de razón - menor     */
   /*         &8   *CHAR     49      8  Nombre de localidad remota  */
   /*         &9   *CHAR     57  *VARY  Unidad de identificador de  */
   /*                                   trabajo                     */
   /*                                                               */
   /*****************************************************************/

             PGM

             DCL        &MSGID   *CHAR LEN(  7)
             DCL        &MSGDTA  *CHAR LEN(100)
             DCL        &MSG     *CHAR LEN(132)

             DCL        &DEVICE  *CHAR LEN( 10)
             DCL        &MODE    *CHAR LEN(  8)
             DCL        &RMTLOC  *CHAR LEN(  8)

             MONMSG     CPF0000  EXEC(GOTO PROBLEM)

           /*********************************************************/
           /* Traer mensajes de cola de mensajes QSYSMSG            */
           /*********************************************************/

   LOOP:     RCVMSG     MSGQ(QSYS/QSYSMSG) WAIT(*MAX) MSGID(&MSGID) +
                           MSG(&MSG) MSGDTA(&MSGDTA)

             IF  ((&MSGID *EQ 'CPF1269') /* msg de arranque fallado */ +
               *AND      ((%BIN(&MSGDTA 45 2) *EQ 704)               +
                 *OR      (%BIN(&MSGDTA 45 2) *EQ 705)) )            +
             THEN(DO)
               /*******************************************************/
               /* Informar sobre error de seguridad a QSECOFR         */
               /*******************************************************/

               CHGVAR &DEVICE %SST(&MSGDTA 1 10) /* Extraer dispositivo */
               CHGVAR &MODE   %SST(&MSGDTA 11 8) /* Extraer modo        */
               CHGVAR RMTLOC %SST(&MSGDTA 49 8)  /* Obtener nombre loc  */

               ENDMOD     RMTLOCNAME(&RMTLOC) MODE(&MODE)

               SNDPGMMSG  MSGID(&MSGID) MSGF(QCPFMSG) MSGDTA(&MSGDTA) +
```

```
                TOMSGQ(QSECOFR)

  SNDPGMMSG  MSG('Dispositivo' *CAT &DEVICE *TCAT ' Modo '  +
               *CAT &MODE *TCAT ' tuvo fallo de seguridad, +
               máx. sesiones cambiado a cero')            +
               TOMSGQ(QSYSOPR)
ENDDO
ELSE DO
  /*********************************************************/
  /* Otro mensaje - Volver a enviar a QSYSOPR              */
  /*********************************************************/

  SNDPGMMSG  MSGID(&MSGID) MSGF(QCPFMSG) MSGDTA(&MSGDTA) +
               TOMSGQ(QSYSOPR)

  /* SNDPGMMSG fallará si el mensaje no tiene un         */
  /*   MSGID o no se encuentra en QCPFMSG                */

  MONMSG     MSGID(CPF0000)  +
             EXEC(SNDPGMMSG MSG(&MSG) TOMSGQ(QSYSOPR))
ENDDO

GOTO       LOOP    /* Ir a buscar mensaje siguiente       */


/*************************************************************/
/* Informar a QSYSOPR de finalización anómala                */
/*************************************************************/

PROBLEM:  SNDPGMMSG  MSG('QSYSMSG ha finalizado de forma anómala') +
                TOMSGQ(QSYSOPR)
          MONMSG     CPF0000

          SNDPGMMSG  MSGID(CPF9898) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
                        MSGDTA('Se ha producido un error inesperado')
          MONMSG     CPF0000

          ENDPGM
```

8.6 Utilización de la Lista de Respuestas del Sistema

La lista de respuestas del sistema le permite especificar que el sistema emita la respuesta a unos mensajes de consulta predefinidos especificados previamente de manera que el usuario de la estación de pantalla no necesite responder. Sólo se pueden responder automáticamente los mensajes de consulta.

La lista de respuestas del sistema contiene identificadores de mensaje, datos de comparación opcionales, un valor de respuesta para cada mensaje y un atributo de vuelco. La lista de respuestas del sistema se aplica solamente a los mensajes de consulta predefinidos que envía un trabajo que utiliza la lista de respuestas del sistema. Se especifica que un trabajo va a utilizar la lista de respuestas del sistema para los mensajes de consulta en el parámetro INQMSGRPY(*SYSRPYL) en los mandatos siguientes:

- Trabajo por Lotes (BCHJOB)
- Someter Trabajo (SBMJOB)
- Cambiar Trabajo (CHGJOB)
- Crear Descripción de Trabajo (CRTJOBD)
- Cambiar Descripción de Trabajo (CHGJOBD)

Cuando un trabajo que utiliza la lista de respuestas del sistema envía un mensaje de consulta predefinido, el sistema busca en la lista de respuestas por orden de número de secuencia ascendente una entrada que coincida con el identificador del mensaje y, opcionalmente, con los datos de comparación del mensaje de respuesta. Si encuentra una entrada, se emite la respuesta especificada y no es necesario que el usuario entre una respuesta. Si no encuentra una entrada, el mensaje se envía al usuario de la estación de pantalla en el caso de los trabajos interactivos o al operador del sistema en el caso de los trabajos por lotes.

La lista de respuestas del sistema se envía con el sistema con las siguientes entradas iniciales definidas:

Número de secuencia	**Identificador de mensaje**	**Valor de comparación**	**Respuesta**	**Vuelco**
10	CPA0700	*NONE	D	*YES
20	RPG0000	*NONE	D	*YES
30	CBE0000	*NONE	D	*YES
40	PLI0000	*NONE	D	*YES

Estas entradas indican que ha de enviarse una respuesta D y que va a tomarse un vuelco del trabajo si un trabajo que utiliza la lista de respuestas del sistema envía el mensaje CPA0700-CPA0799, RPG0000-RPG9999, CBE0000-CBE9999 ó PLI0000-PLI9999 (que indican una anomalía en el programa). Para que el sistema utilice estas entradas, hay que especificar que los trabajos han de utilizar la lista de respuestas del sistema.

Para añadir otros mensajes de consulta a la lista de respuestas del sistema, utilice el mandato Añadir Entrada en Lista de Respuestas (ADDRPYLE). En este mandato, puede especificar el número de secuencia, el identificador de mensaje, los datos de comparación opcionales, el CCSID de los datos de comparación, la acción de respuesta y el atributo de vuelco. Se puede acceder fácilmente a la función del mandato ADDRPYLE mediante el mandato Trabajar con Entradas de Lista de Respuestas del Sistema (WRKRPYLE).

Las acciones de respuesta siguientes pueden especificarse para los mensajes de consulta que se colocan en la lista de respuestas del sistema (el valor del parámetro aparece entre paréntesis):

- Enviar la respuesta por omisión a los mensajes de consulta (*DFT). En este caso, se envía la respuesta por omisión del mensaje. El mensaje no se visualiza y no se llama a ningún programa de manejo por omisión.

- Pedir al usuario de la estación de trabajo o al operador del sistema que respondan al mensaje (*RQD). Si la cola de mensajes a la que se envía el mensaje (cola de mensajes de la estación de trabajo para trabajos interactivos y QSYSOPR para trabajos por lotes) está en modalidad de interrupción, el mensaje se visualiza y el usuario de la estación de trabajo debe responder al mensaje. Esta opción funciona como si la lista de respuestas del sistema no se utilizara.

- Enviar la respuesta especificada en la entrada de la lista de respuestas del sistema (respuesta del mensaje, 32 caracteres como máximo). En este caso, la respuesta especificada se envía como respuesta al mensaje. El mensaje no se visualiza y no se llama a

ningún programa de manejo por omisión.

Los mandatos siguientes añaden entradas a la lista de respuestas del sistema para los mensajes RPG1241, RPG1200, CPA4002, CPA5316 y cualquier otro mensaje de consulta:

```
   ADDRPYLE SEQNBR(15) MSGID(RPG1241) RPY(C)
   ADDRPYLE SEQNBR(18) MSGID(RPG1200) RPY(*DFT) DUMP(*YES)
   ADDRPYLE SEQNBR(22) MSGID(CPA4002) RPY(*RQD) +
              CMPDTA('QSYSPRT')
   ADDRPYLE SEQNBR(25) MSGID(CPA4002) RPY(G)
   ADDRPYLE SEQNBR(27) MSGID(CPA5316) RPY(I) DUMP(*NO) +
              CMPDTA('QSYSPRT' 21)
   ADDRPYLE SEQNBR(9999) MSGID(*ANY) RPY(*DFT)
```

La lista de respuestas del sistema aparece ahora así:

Número de secuencia	Identific. de mensaje	Valor de comparación (b=blanco)	Posición inicial de compa- ración	Respuesta	Vuelco
10	CPA0700		1	D	*YES
15	RPG1241		1	C	*NO
18	RPG1200		1	*DFT	*YES
20	RPG0000		1	D	*YES
22	CPA4002	**'QSYSPRT'**	1	*RQD	*NO
25	CPA4002		1	G	*NO
27	CPA5316	**'QSYSPRT'**	21	I	*NO
30	CBE0000		1	D	*YES
40	PLI0000		1	D	*YES
9999	*ANY		1	*DFT	*NO

Para un trabajo que utiliza esta lista de respuestas del sistema, se produce lo siguiente cuando el trabajo envía los mensajes que se han añadido a la lista de respuestas:

- Para el número de secuencia 15, cada vez que un trabajo que utiliza la lista de respuestas del sistema envía un mensaje RPG1241, se envía una respuesta C y el trabajo no se vuelca.

- Para el número de secuencia 18, se utiliza un identificador genérico de mensaje para que, siempre que el trabajo envíe un mensaje de consulta RPG1200, se envíe la respuesta por omisión. Ésta puede ser la respuesta por omisión especificada en la descripción de mensaje o la del sistema. Antes de enviar la respuesta por omisión, se vuelca el trabajo. La entrada que se añadió anteriormente altera temporalmente esta entrada para el mensaje RPG1241.

- Para el número de secuencia 22, si se envía el mensaje de consulta CPA4002 con los datos de comparación de QSYSPRT, el mensaje se envía al usuario de la estación de pantalla, quien debe emitir una respuesta.

  Cuando se especifica un valor de comparación sin una posición inicial, dicho valor se compara con los datos del mensaje comenzando por la posición 1 de los datos de substitución del mensaje.

  El número de secuencia 22 comprueba si hay un nombre de dispositivo de impresora que sea QSYSPRT. Para un ejemplo de comprobación de una variable de sustitución con una posición inicial diferente, véase el número de secuencia 27.

- Para el número de secuencia 25, si el mensaje de consulta CPA4002 (verificar alineación en la impresora &1.) se envía con la comparación distinta de QSYSPRT, se envía una respuesta de G, y el trabajo no se vuelca. El número de secuencia 22 precisa una respuesta del operador al mensaje de alineación de formularios si el dispositivo de impresión es QSYSPRT. El número de secuencia 25 define que si el mensaje de consulta de alineación de formularios se da para cualquier otro dispositivo, se supone por omisión una respuesta de G=Continuar.

- Para un número de secuencia 27, si el mensaje de consulta CPA5316 se envía con los datos de comparación **TESTEDFILETSTLIBRARYQSYSPRT**, se envía una respuesta **I**.

  Cuando se especifican un valor de comparación y una posición inicial, el valor de comparación se compara con los datos del mensaje empezando con la posición inicial. En tal caso, la posición 21 es el comienzo de la tercera variable de sustitución. Para el mensaje CPA5316, las cuatro primeras variables de sustitución son las siguientes:

&1	Nombre de archivo ODP	*CHAR	10
&2	Nombre de biblioteca ODP	*CHAR	10
&3	Nombre de dispositivo ODP	*CHAR	10
&4	Número de línea para la primera línea	*BIN	2

  Por lo tanto, el número de secuencia 27 comprueba si hay un nombre de dispositivo ODP que sea QSYSPRT antes de enviar una respuesta.

- Para un número de secuencia **9999**, el identificador de mensaje de *ANY se aplica a cualquier mensaje de consulta predefinido que no coincida con ninguna entrada con un número de secuencia más bajo y se envía la respuesta por omisión de estos mensajes de consulta. Si esta entrada no estuviera incluida en la lista de respuestas del sistema, el usuario de la estación de pantalla tendría que responder a todos los demás mensajes de consulta que no estuvieran incluidos en la lista de respuestas del sistema.

Cuando el valor de comparación contiene datos *CCHAR, los datos del mensaje de la función emisora se convierten al CCSID de los datos de mensaje almacenados en la lista de respuestas del sistema antes de realizar la comparación. Sólo se convierten datos de tipo *CCHAR. Consulte el mandato Añadir Descripción de Mensaje (ADDMSGD) del manual CL Reference para obtener más información acerca de los datos *CCHAR.

PRECAUCIÓN:
Al utilizar datos *CCHAR como datos de comparación, se aplican las siguientes restricciones:

- No puede mezclar datos de tipo *CCHAR con otros datos cuando se añade este tipo de entrada de lista de respuestas.

- No puede incluir la longitud de los datos *CCHAR en los datos de comparación.

Si combina datos *CCHAR o incluye la longitud de los datos *CCHAR, pueden producirse resultados imprevisibles.

Una entrada permanece en la lista de respuestas del sistema hasta que se utiliza el mandato Eliminar Entrada en Lista de Respuestas (RMVRPYLE) para suprimirla. Puede utilizarse el mandato Cambiar Entrada en Lista de Respuestas (CHGRPYLE) para cambiar los atributos de una entrada de la lista de respuestas y el mandato Trabajar con Entrada en Lista de Respuestas del Sistema (WRKRPYLE) para visualizar las entradas de respuesta que hay actualmente en la lista de respuestas.

8.7 Anotación de Mensajes

Los dos tipos de anotaciones para mensajes son:

- Anotaciones de trabajo
- Anotaciones históricas

Las anotaciones de trabajo contienen información relacionada con las peticiones entradas para un trabajo. Las anotaciones de QHST contienen datos del sistema, tales como la historia de un arranque de trabajo y el final de la actividad en el sistema.

Subtemas

8.7.1 Anotaciones de trabajo

Cada trabajo tiene unas anotaciones de trabajo asociadas que pueden contener lo siguiente:

- Los mandatos del trabajo.
- Los mandatos de un programa CL si éste se creó con la opción LOG(*YES) o con la opción LOG(*JOB), y se ejecutó un mandato Cambiar Trabajo (CHGJOB) con la opción LOGCLPGM(*YES) (para más información sobre anotaciones de mandatos de un programa CL, véase "Anotación de Mandatos de Procedimiento CL" en el tema 2.7.1).
- Todos los mensajes y ayudas de mensaje enviados al peticionario y que no se han eliminado de las colas de mensajes de llamada.

Una vez finalizado el trabajo, pueden grabarse las anotaciones de trabajo en el archivo de salida QPJOBLOG o en un archivo de base de datos. Desde el archivo de salida QPJOBLOG, pueden imprimirse las anotaciones de trabajo; desde un archivo de base de datos, puede consultarse la información de anotaciones de trabajo utilizando una base de datos. También puede especificar que no se graben las anotaciones de trabajo para trabajos ejecutados satisfactoriamente--más adelante en este mismo capítulo se ofrece una explicación de cómo evitar las anotaciones de trabajo.

Para grabar las anotaciones de trabajo en una base de datos, se precisa la utilización de la API QMHCTLJL--consulte el manual System API Reference para obtener más información acerca de las API. Cuando las anotaciones de trabajo se direccionan a la base de datos, pueden generarse uno o dos archivos. El archivo primario contiene la información principal de un mensaje, como ID del mensaje, gravedad del mensaje, tipo de mensaje y datos del mensaje. El archivo secundario contiene las imágenes de impresión del texto del mensaje. El segundo archivo es opcional y su producción la controlan los parámetros de la API QMHCTLJL. Ambos archivos se describen externamente y se pueden procesar utilizando dispositivos de consulta y base de datos del sistema. Consulte Apéndice C, "Archivos de Salida de Anotaciones de Trabajo" en el tema C.0 para obtener información referente a los formatos de los archivos primarios y secundarios.

Puede controlar qué información se graba en las anotaciones de trabajo. Para ello, especifique el parámetro LOG en el mandato Crear Descripción de Trabajo (CRTJOBD). Puede cambiar estos niveles utilizando el mandato Cambiar Trabajo (CHGJOB) o Cambiar Descripción de Trabajo (CHGJOBD). El parámetro LOG consta de 3 valores: nivel de mensaje, gravedad de mensaje y nivel de texto de mensaje. Para obtener más información acerca de estos mandatos, consulte el manual CL Reference.

El primer valor, el nivel de mensaje, tiene los siguientes niveles:

Nivel Descripción

0 No se anotan datos.

1 La única información a anotar es la de todos los mensajes enviados a la cola externa de mensajes del trabajo con una gravedad superior o igual a la gravedad especificada del mensaje. Los mensajes de este tipo indican cuando se arrancó el trabajo, cuando finalizó y su estado al terminar.

2 Se anota la siguiente información:

- Información sobre anotación del nivel 1.
- Las peticiones entradas en una línea de mandatos o los mandatos que se anotan desde un programa CL para el que se emiten mensajes de alto nivel cuya gravedad es superior o igual a la gravedad especificada. Si se anota la petición o el mandato, también se anotan todos los mensajes asociados.

3 Se anota la siguiente información:

- Información sobre anotación del nivel 1.
- Todas las peticiones o mandatos que están anotándose desde un programa CL.
- Si cualquier mensaje de alto nivel asociado con una petición o mandato tiene una gravedad superior o igual a la gravedad especificada, entonces se anotan todos los mensajes asociados con la petición o mandato.

4 Se anota la siguiente información:

- Todas las peticiones o mandatos que están anotándose desde un programa CL.
- Solamente los mensajes con un código de gravedad superior o igual a la gravedad especificada.

Nota: Un mensaje de alto nivel es el que se envía a la cola de mensajes CALL de programa del programa que recibe la petición o mandato que está anotándose desde un programa CL.

El segundo valor, gravedad del mensaje, indica el nivel mínimo de gravedad que hace que entren mensajes de error en las anotaciones de trabajo. Se permiten valores comprendidos entre 0 y 99. Sólo aquellos mensajes con una gravedad mayor o igual a este valor se pueden introducir en las anotaciones de trabajo.

El tercer valor del parámetro LOG, el nivel de texto de mensaje, especifica el nivel de texto de mensaje que se graba en las anotaciones de trabajo. Los valores son:

***MSG** Sólo el texto del mensaje se graba en las anotaciones de trabajo (no se incluye la ayuda del mensaje).

***SECLVL** El mensaje y la ayuda correspondiente se graban en las anotaciones de trabajo.

***NOLIST** Si el trabajo finaliza con normalidad, no se producen anotaciones de trabajo. Si el código de fin de trabajo es 20 o más, se producen anotaciones de trabajo en las que se incluyen el mensaje y la ayuda del mensaje.

Antes de que un programa de proceso de peticiones reciba una petición nueva, se produce un filtrado del mensaje. El **filtrado de mensajes** es el proceso de eliminación de mensajes de las anotaciones de trabajo en función del nivel de anotaciones de mensajes establecido para el trabajo.

El filtrado no tiene lugar después de llamar a cualquier mandato CL en un programa. Por este motivo, si se ejecuta interactivamente un programa CL o se somete a un proceso por lotes, la filtración se ejecuta una vez después de que el programa finaliza, ya que el programa no es procesador de peticiones.

Nota: Puesto que *NOLIST especifica que no se colocan en spool anotaciones de trabajo correspondientes a trabajos que finalizan con normalidad, la eliminación de los mensajes de estas anotaciones especificando un nivel de anotaciones 0 supone un uso innecesario de los recursos del sistema en trabajos por lotes.

En el ejemplo siguiente se muestra cómo afecta el nivel de anotaciones a la información que está almacenada en la cola de mensajes de trabajo y que, por lo tanto, se graba en las anotaciones de trabajo, si se solicita. Este ejemplo también muestra que, cuando el mandatos se ejecuta de forma interactiva, el filtrado tiene lugar después de cada mandato.

Nota: En los ejemplos se incluyen los niveles de mensaje de alto nivel y detallado. Los mensajes de alto nivel están identificados como *Mensaje*; los mensajes detallados se identifican como *Mensajes Detallado*.

1. El mandato CHGJOB especifica un nivel de anotación de 2, una gravedad de mensaje de 50 y que solamente se graban mensajes en las anotaciones de trabajo (*MSG).

```
+------------------------------------------------------------------------------+
|                                                                              |
|                              Entrada de Mandatos                  SYSTEM1    |
|                                                  Nivel petición:   1         |
| Mandatos y mensajes anteriores:                                              |
|      > CHGJOB LOG(2 50 *MSG)                                                 |
|                                                                              |
|                                                                              |
```

2. PGMA envía tres mensajes informativos con códigos de gravedad 20, 50 y 60 a su propia cola de mensajes de llamada y a la cola de mensajes del programa llamada antes de la llamada especificada (*PRV). Los mensajes que envía PGMA a su propia cola de mensajes de llamada se llaman mensajes detallados. Los **Mensajes detallados** son aquellos

mensajes que se envían a la cola de mensajes de llamada, de la llamada de programa de nivel inferior.

PGMB envía dos mensajes informativos con códigos de gravedad 40 y 50 a su propia cola de mensajes de llamada. Estos son mensajes detallados. PGMB envía también un mensaje informativo con un código de gravedad 10 a *PRV.

Observe que el mandato CHGJOB ya no aparece en la pantalla. De acuerdo con el nivel de anotación 2, solamente las peticiones para las cuales se ha emitido un mensaje con una gravedad igual o superior a la especificada se salvan para las anotaciones de trabajo y no se emiten mensajes para dicha petición. Si dicho mensaje se hubiera emitido, cualquier mensaje detallado que se hubiera emitido se salvaría para las anotaciones de trabajo y podría visualizarse pulsando F10.

```
                              Entrada de Mandatos                    SYSTEM1
                                                    Nivel petición:   1
Mandatos y mensajes anteriores:
      > CALL PGMA
        Gravedad mensaje 20 - PGMA
        Gravedad mensaje 50 - PGMA
        Gravedad mensaje 60 - PGMA
      > CALL PGMB
        Gravedad mensaje 10 - PGMB









                                                                      Final
Teclee mandato, pulse Intro.
===> ____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
 F3=Salir  F4=Solicitud   F9=Recuperar  F10=Incluir mensajes detallados
 F11=Visualizar completo  F12=Cancelar  F13=Information Assistant  F24=Más teclas
```

3. Cuando se pulsa F10=Incluir mensajes detallados desde la pantalla Entrada de Mandatos, se visualizan todos los mensajes asociados a la solicitud. Pulse F10 otra vez para excluir los mensajes detallados.

```
                              Entrada de Mandatos                    SYSTEM1
                                                    Nivel petición:   1
Todos los mandatos y mensajes anteriores:
   > CALL PGMA
     Gravedad msj detallado 20 - PGMA
     Gravedad msj detallado 50 - PGMA
     Gravedad msj detallado 60 - PGMA
     Gravedad mensaje 20 - PGMA
     Gravedad mensaje 50 - PGMA
     Gravedad mensaje 60 - PGMA
   > CALL PGMB
     Gravedad msj detallado 40 - PGMB
     Gravedad msj detallado 50 - PGMB
     Gravedad mensaje 10 - PGMB

                                                                      Final
Teclee mandato, pulse Intro.
===> ____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
 F3=Salir            F4=Solicitud  F9=Recuperar  F10=Excluir mensajes detallados
 F11=Visualizar compl  F12=Cancelar    F13=Information Assistant  F24=Más teclas
```

4. Cuando se entra otro mandato (en este ejemplo, CHGJOB), se eliminan el mandato **CALL PGMB** y todos los mensajes (incluidos los mensajes detallados), Se eliminan, ya que el código de gravedad para el mensaje de alto nivel asociado con esta petición no era igual ni mayor que el código de gravedad especificado en el mandato CHGJOB. El mandato **CALL PGMA** y sus mensajes asociados permanecen porque como mínimo uno de los

mensajes de alto nivel emitidos para dicha petición tiene un código de gravedad igual o superior al especificado.

En la pantalla siguiente, el mandato CHGJOB especifica un nivel de anotación de 3, una gravedad de mensaje de 40 y que el texto de mensaje de primer y de segundo nivel se graben en las anotaciones de trabajo. Cuando se entra otro mandato, el mandato CHGJOB permanece en la pantalla porque el nivel de anotación 3 anota todas las peticiones.

PGMC envía dos mensajes con códigos de gravedad 30 y 40 a la cola de mensajes de llamada del programa llamado antes de la llamada especificada (*PRV).

PGMD envía un mensaje con una gravedad de 10 a *PRV.

```
                              Entrada de Mandatos                    SYSTEM1
                                                        Nivel petición:   1
Mandatos y mensajes anteriores:
     > CALL PGMA
       Gravedad mensaje 20 - PGMA
       Gravedad mensaje 50 - PGMA
       Gravedad mensaje 60 - PGMA
     > CHGJOB LOG(3 40 *SECLVL)
     > CALL PGMC
       Gravedad mensaje 30 - PGMC
       Gravedad mensaje 40 - PGMC
     > CALL PGMD
       Gravedad mensaje 10 - PGMD


                                                                      Final
Teclee mandato, pulse Intro.
===> ____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
 F3=Salir  F4=Solicitud F9=Recuperar  F10=Incluir mensajes detallados
 F11=Visualizar compl  F12=Cancelar  F13=Information Assistant  F24=Más teclas
```

5. Cuando se entra otro mandato después de entrar el mandato **CALL PGMD**, éste permanece en la pantalla, pero se suprime el mensaje que tiene asociado, ya que su código de gravedad no es igual ni superior al código de gravedad especificado en el parámetros LOG del mandato CHGJOB.

   Se entra el mandato **SIGNOFF *LIST** para imprimir las anotaciones de trabajo.

```
                              Entrada de Mandatos                    SYSTEM1
                                                        Nivel petición:   1
Mandatos y mensajes anteriores:
     > CHGJOB LOG(3 40 *SECLVL)
     > CALL PGMC
       Gravedad mensaje 30 - PGMC
       Gravedad mensaje 40 - PGMC
     > CALL PGMD
     > CALL PGME









                                                                      Final
Teclee mandato, pulse Intro.
===> SIGNOFF *LIST__________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
 F3=Salir  F4=Solicitud F9=Recuperar  F10=Incluir mensajes detallados
 F11=Visualizar compl  F12=Cancelar  F13=Information assistant  F24=Más teclas
```

Las anotaciones de trabajo, que aparecen a continuación, contienen todas

las peticiones y todos los mensajes que han permanecido en la pantalla Entrada de Mandatos. Además, las anotaciones de trabajo contienen la ayuda de mensaje asociada con cada mensaje, tal como lo especifica el mandato CHGJOB. Observe que las anotaciones de trabajo contienen la ayuda de mensaje de todos los mensajes emitidos durante el trabajo, no solamente de los mensajes emitidos desde que se introdujo el segundo mandato CHGJOB.

```
 5763SS1 V2R3M0 930925                      Anotaciones de Trabajo              SYSAS727 12/12/92 07
  Nombre de trabajo  . . . . . :   QPADEV0007      Usuario . . . . . :   JOHNDOE      Número . . . . :
  Descripción trabajo. . . . . :   QDFTJOBD        Biblioteca  . . . :   QGPL
IDMSJ      TIPO                   GRAV  FECHA      HORA       DE PGM         BIBLIOTECA  INST     A PGM
CPF1124    Información            00    12/12/93   07:57:16   QWTPIIPP       QSYS        04FC     *EXT
                                    Mensaje . . . . :   Trabajo 004201/JOHNDOE/QPADEV0007 arrancado el 1
                                      07:57:16 en subsistema QINTER en QSYS. Trab entrado sistema el 12/
                                      07:57:16.
*NONE      Petición                     12/12/93   07:57:50   QMHGSD         QSYS        0322     QCMD
                                    Mensaje . . . . :  -call pgma
CPF1001    Información            20    12/12/93   07:57:50   PGMA           JOHNDOE     000C     PGMA
                                    Mensaje . . . . :   Gravedad msj detall  20 - PGMA
                                    CPF1001 texto seg nivel - PGMA
CPF1002    Información            50    12/12/93   07:57:50   PGMA           JOHNDOE     0010     PGMA
                                    Mensaje . . . . :   Gravedad msj detall  50 - PGMA
                                    CPF1002 texto seg nivel - PGMA
CPF1003    Información            60    12/12/93   07:57:50   PGMA           JOHNDOE     0014     PGMA
                                    Mensaje . . . . :   Gravedad msj detall 60 - PGMA
                                    CPF1003 texto seg nivel - PGMA
CPF1004    Información            20    12/12/93   07:57:50   PGMA           JOHNDOE     0018     QCMD
                                    Mensaje . . . . :   Gravedad msj 20 - PGMA
                                    CPF1004 texto seg nivel - PGMA
CPF1005    Información            50    12/12/93   07:57:50   PGMA           JOHNDOE     001C     QCMD
                                    Mensaje . . . . :   Gravedad msj 50 - PGMA
                                    CPF1005 texto seg nivel - PGMA
CPF1006    Información            60    12/12/93   07:57:50   PGMA           JOHNDOE     0020     QCMD
                                    Mensaje . . . . :   Gravedad msj 60 - PGMA
                                    CPF1006 texto seg nivel - PGMA
*NONE      Petición                     12/12/93   07:58:31   QMHGSD         QSYS        0322     QCMD
                                    Mensaje . . . . :  -chgjob log(3 40 *seclvl)
*NONE      Petición                     12/12/93   07:58:34   QMHGSD         QSYS        0322     QCMD
                                    Mensaje . . . . :  -call pgmc
CPF100F    Información            30    12/12/93   07:58:34   PGMC           JOHNDOE     000C     QCMD
                                    Mensaje . . . . :   Gravedad msj 30 - PGMC
                                    CPF100F texto seg nivel - PGMC
CPF1010    Información            40    12/12/93   07:58:34   PGMC           JOHNDOE     0010     QCMD
                                    Mensaje . . . . :   Gravedad msj 40 - PGMC
                                    CPF1010 texto seg nivel - PGMC
*NONE      Petición                     12/12/93   07:58:38   QMHGSD         QSYS        0322     QCMD
                                    Mensaje . . . . :  -call pgmd
*NONE      Petición                     12/12/93   07:58:45   QMHGSD         QSYS        0322     QCMD
                                    Mensaje . . . . :  -call pgme
*NONE      Petición                     12/12/93   07:58:52   QMHGSD         QSYS        0322     QCMD
                                    Mensaje . . . . :  -signoff *list
CPF1164    Terminación            00    12/12/93   07:58:52   QWTMCEOJ       QSYS        01EE     *EXT
                                    Mensaje . . . . :   trabajo 004201/JOHNDOE/QPADEV0007 finalizado el
                                      07:58:52; 3 segund utilizados; cód finalización 0.
                                    Causa . . . . . :   Trabajo 004201/JOHNDOE/QPADEV0007 terminado el 1
                                      07:58:52 tras utilizar 3 seg tiempo unidad de proceso. El trab tie
                                      finaliz 0. El trab finalizó tras 1 paso direcc con cód finaliz sec
                                      0. Los cód finaliz y sus significados son:  0 - El trabajo se ha c
                                      normalmente. 10 - El trabajo se ha completado normalm durante fin
                                      o fin subsistema controlado. 20 - El trab sobrepasó gravedad de fi
                                      trab ENDSEV). 30 - El trab finalizó anormalmente. 40 - El trab fin
                                      estar activo. 50 - El trab finalizó mientras el trab estaba activo
                                      subsistema finalizó anormalmente mientras el trab estaba activo. 7
                                      anormalmente mientras el trab estaba activo. 80 - El trab finalizó
                                      90 - Se forzó el trab a finalizar tras límite tiempo (mandato ENDJ
                                      Recuperación  . :   Para más información, vea Gestión de Trabajos,
                                      SC41-8078.
```

Las cabeceras situadas al principio de cada página de las anotaciones de trabajo impresas identifican el trabajo al que se aplican las anotaciones de trabajo y las características de cada entrada:

- El nombre calificado al completo del trabajo (nombre de trabajo, nombre de usuario y número de trabajo).
- El nombre de la descripción de trabajo utilizada para arrancar el trabajo.
- La fecha y la hora en que se arrancó el trabajo.
- El identificador del mensaje.

- El tipo de mensaje.
- La gravedad del mensaje.
- La fecha y la hora en que se envió cada mensaje. No está incluido en los mensajes de petición.
- El mensaje. Si el nivel de anotación especifica que se va a incluir texto de segundo nivel, dicho texto aparece en las líneas situadas debajo del mensaje.
- El programa o procedimiento desde el que se envió el mensaje o la petición.
- El número de instrucción de la interfaz de máquina o número de sentencia de alto nivel del programa o el procedimiento desde el que se envió el mensaje.
- El programa o procedimiento al que se envió el mensaje o la petición.
- El número de instrucción de la interfaz de máquina o número de sentencia de lenguaje de alto nivel a la que se ha enviado el programa o procedimiento.

Subtemas

8.7.1.1 Envío o Recepción de un Programa o Procedimiento

Cuando el emisor o el receptor es un procedimiento ILE, la entrada de mensaje contiene el nombre completo del procedimiento (nombre del procedimiento, nombre de módulo y nombre del programa ILE). Cuando el emisor o el receptor es un programa OPM (modelo de programa original), sólo se muestra el nombre del programa OPM.

|Si el remitente o receptor es un programa OPM, el número de instrucción
|correspondiente representa un número de instrucción. Sólo existe uno. Si el emisor o el receptor es un procedimiento ILE, el número de instrucción representa un número de sentencia de lenguaje de alto nivel en lugar de un número de instrucción MI. Si el procedimiento ILE se ha mejorado (eficacia máxima), puede haber un máximo de tres números. No siempre es posible determinar un sólo número de sentencia para un procedimiento mejorado. Si se da más de un número, cada uno de ellos representa un punto potencial en el que se encontraba el procedimiento cuando se envió el mensaje. También es posible que no pueda determinarse ningún número. En este caso, en lugar de un número aparecerá *N en el mensaje.

Los niveles de anotación afectan a unas anotaciones de trabajo por lotes de la misma forma que se muestra en el ejemplo anterior. Si el trabajo utiliza APPC, la cabecera contiene una línea que muestra la unidad del identificador de trabajo para APPC. Consulte el manual APPC Programming para obtener más información acerca de APPC.

8.7.1.2 Filtrado Adicional de Mensajes

Si las anotaciones de trabajo se dirigen a un archivo de base de datos utilizando la API QMHCTLJL, puede especificarse el filtraje adicional de mensajes. El filtraje de mensajes especificado a través de esta API se aplica cuando finaliza el trabajo y los registros para los mensajes se graban en el archivo. Hasta este momento, han aparecido los mensajes filtrados. Así, pueden visualizarse mientras se está ejecutando el trabajo. Cuando se graban las anotaciones de trabajo, los mensajes que se filtran no tendrán registros grabados en el archivo para los mismos. Así, aunque aparezcan mientras se está ejecutando el trabajo, no aparecerán en el archivo final que se produce.

8.7.1.3 Visualización de las Anotaciones de Trabajo

La forma en que se visualizan las anotaciones de trabajo depende del estado de éste.

- Si el trabajo ha finalizado y las anotaciones del mismo aún no se han impreso, utilice el mandato Visualizar Archivo en Spool (DSPSPLF), tal como se indica a continuación:

  ```
  DSPSPLF FILE(QPJOBLOG) JOB(001293/FRED/WS3)
  ```

  para visualizar las anotaciones de trabajo correspondientes al trabajo número 001293 asociado con el usuario FRED en la estación de pantalla WS3.

- Si al trabajo aún está activo (interactivo o por lotes) o está en una cola de trabajos y todavía no se ha arrancado, utilice el mandato Visualizar Anotaciones de Trabajo (DSPJOBLOG). Por ejemplo, para visualizar las anotaciones de trabajo del trabajo interactivo del usuario JSMITH en la estación de pantalla WS1, entre:

  ```
  DSPJOBLOG JOB(nnnnnn/JSMITH/WS1)
  ```

  siendo **nnnnnn** el número del trabajo.

Para visualizar las anotaciones de trabajo de su propio trabajo interactivo, efectúe una de las acciones siguientes:

- Entre el mandato siguiente:

  ```
  DSPJOBLOG
  ```

- Entre el mandato WRKJOB y seleccione la opción 10 (Visualizar anotaciones de trabajo) de la pantalla Trabajar con Trabajo.

- Pulse F10=Incluir mensajes detallados de la pantalla Entrada de Mandatos (esta tecla visualiza los mensajes que se muestran en las anotaciones de trabajo).

- Si la luz de entrada inhibida permanece encendida en su estación de pantalla, haga lo siguiente:

  1. Pulse la tecla Petición de Sistema y a continuación pulse la tecla Intro.
  2. En el menú de Petición de Sistema, seleccione la opción 3 (Visualizar trabajo actual).
  3. En el menú de Visualizar Trabajo, seleccione la opción 10 (Visualizar Anotaciones de Trabajo si está activo o en la cola de trabajos).
  4. En la pantalla Anotaciones de Trabajo, DSPJOB aparece como la petición que se procesa. Pulse F10 (Visualizar mensajes detallados).
  5. En la pantalla Visualizar Todos los Mensajes, pulse la tecla Giro Abajo para ver los mensajes que se recibieron antes de pulsar la tecla Petición del Sistema.

- Desconecte la estación de trabajo, especificando LOG(*LIST) en el mandato SIGNOFF.

Cuando utilice el mandato Visualizar Anotaciones de Trabajo (DSPJOBLOG), verá la pantalla Anotaciones de Trabajo. Dicha pantalla muestra nombres de programas con los símbolos especiales siguientes:

>> El mandato en ejecución o el próximo mandato a ejecutar. Por ejemplo, si se ha llamado a un programa, se muestra la llamada al programa.

> El mandato ha finalizado el proceso.

. . El mandato todavía no ha sido procesado.

? Mensaje de respuesta. Este símbolo marca tanto los mensajes que necesitan una respuesta como los que han sido contestados.

En la pantalla Anotaciones de Trabajo, puede hacerse lo siguiente:

- Pulsar F10 para visualizar los mensajes detallados. Este pantalla muestra los mandatos u operaciones que se ejecutaron en un programa HLL o en un programa o procedimiento CL para los cuales LOG está activado.

- Utilizar las teclas de movimiento del cursor para ir al final de las anotaciones de trabajo. Para desplazarse rápidamente, pulse F18 (Final). Después de pulsar F18, puede ser necesario girar hacia abajo para ver el mandato que se está ejecutando.

- Utilizar las teclas de movimiento del cursor para ir al principio de las anotaciones de trabajo. Para desplazarse rápidamente, pulse (Principio).

Debe utilizar el mandato DSPJOBLOG para dirigir el trabajo a un archivo de base de datos en vez de imprimirlo o visualizarlo. Hay dos opciones disponibles. En la primera opción, puede especificar un nombre de archivo y de miembro en el mandato. En esta opción, la información de anotaciones de trabajo primarias se graban en el archivo de base de datos especificado en el mandato. En la segunda opción puede utilizar el mandato junto con la información proporcionada en la API QMHCTLJL ejecutada anteriormente. En esta opción, las anotaciones de trabajo se graban en el archivo(s) especificado(s) en la llamada de la API. Con esta opción, pueden producirse archivos primarios y secundarios y puede realizarse el filtrado de mensajes a medida que los mensajes se graban en el archivo. Con estas dos opciones, cuando finalice el mandato DSPJOBLOG, la salida no se visualizará ni habrá un archivo en spool disponible para la impresión.

8.7.1.4 Impedir la Producción de Anotaciones de Trabajo

Para impedir que se produzcan anotaciones de trabajo en la terminación de un trabajo de proceso por lotes, puede especificar *NOLIST para el valor de nivel de texto de mensaje del parámetro LOG del mandato Trabajo por Lotes (BCHJOB), Someter Trabajo (SBMJOB), Cambiar Trabajo (CHGJOB), Crear Descripción de Trabajo (CRTJOBD) o Cambiar Descripción de Trabajo (CHGJOBD). Si especifica *NOLIST para el valor del nivel del mensaje del parámetro LOG, no se producen anotaciones de trabajo al final del trabajo a menos que el código de finalización del trabajo sea 20 o más, en cuyo caso sí se producen anotaciones de trabajo.

Para un trabajo interactivo, el valor especificado para el parámetro LOG del mandato SIGNOFF prevalece sobre el valor del parámetro LOG especificado para el trabajo.

8.7.1.5 Consideraciones sobre Anotaciones de Trabajo

Las siguientes sugerencias se aplican a la utilización de anotaciones de trabajo:

- Para cambiar la cola de salida para todos los trabajos del sistema, utilice el parámetro OUTQ o DEV del mandato Cambiar Archivo de Impresora (CHGPRTF) para cambiar el archivo QSYS/QPJOBLOG. Los dos ejemplos siguientes utilizan cada uno de estos parámetros:

  ```
  CHGPRTF FILE(QSYS/QPJOBLOG)
          DEV (USRPRT)
  ```

  o

  ```
  CHGPRTF FILE(QSYS/QPJOBLOG)
          OUTQ(USROUTQ)
  ```

- Para modificar el archivo de impresora QPJOBLOG para que utilice la cola de salida QEZJOBLOG, use la función de borrado de Operational Assistant. Cuando desee utilizar el borrado automático de las anotaciones de trabajo, los archivos de impresora deben dirigirse a esta cola de salida. Para obtener más información sobre la función de borrado de Operational Assistant, véase el manual *Operación del sistema*.

- Para especificar la cola de salida en la que se graban las anotaciones de un trabajo, asegúrese de que en el archivo QPJOBLOG está especificado OUTQ(*JOB). Puede utilizar el parámetro OUTQ en cualquiera de estos mandatos: BCHJOB, CRTJOBD, CHGJOBD o CHGJOB. A continuación se facilita un ejemplo:

  ```
  CHGJOB OUTQ(*JOB)
  ```

  Si cambia OUTQ por omisión al principio del trabajo, todos los archivos en spool resultan afectados. Si lo cambia justo antes de que finalice el trabajo, sólo se ven afectadas las anotaciones de trabajo. No puede utilizar el mandato Alterar Temporalmente con Archivo de Impresora (OVRPRTF) para que afecte a las anotaciones de trabajo.

- Si no se encuentra la cola de salida de un trabajo, no se producen anotaciones de trabajo.

- Para retener todas las anotaciones de trabajo, especifique HOLD(*YES) en el mandato CHGPRTF para el archivo QSYS/QPJOBLOG. Las anotaciones de trabajo se liberan en el transcriptor cuando se ejecuta el mandato Liberar Archivo en Spool (RLSSPLF). A continuación se facilita un ejemplo:

  ```
  CHGPRTF FILE(QSYS/QPJOBLOG)
          HOLD(*YES)
  ```

- Si el sistema finaliza de forma anómala, la solicitud de arranque permite al operador del sistema especificar si las anotaciones de trabajo van a imprimirse para los trabajos que estaban activos en el momento de la finalización anómala.

- Para suprimir unas anotaciones de trabajo, utilice el mandato Suprimir Archivo en Spool (DLTSPLF) o la opción Suprimir de la pantalla de cola de salida.

- Si ha utilizado el parámetro USRDTA en el mandato Cambiar Archivo de Impresión (CHGPRTF) para cambiar el valor de los datos de usuario para el archivo QSYS/QPJOBLOG, el valor especificado no se mostrará en las pantallas Trabajar con Cola de Salida o Trabajar con Todos los Archivos en Spool. El valor mostrado en la columna de datos de usuario es el nombre del trabajo cuyas anotaciones de trabajo se han impreso.

- Si se están analizando las anotaciones de trabajo mediante técnica de programación, utilice la API QMHCTLJL para dirigir las anotaciones de trabajo al archivo(s) de base de datos. El formato de los registros del archivo de base de datos está garantizado, mientras que el formato impreso no lo está. Si deben añadirse nuevos campos en un registro de las anotaciones de trabajo, estos se añaden al final del registro de
| modo que los programas existentes sigan funcionando. Los dispositivos
| de consulta proporcionados por el sistema, se pueden utilizar
| directamente en los archivos.

8.7.1.6 Consideraciones para las Anotaciones de Trabajo Interactivo

Las descripciones de trabajos proporcionadas por IBM QCTL, QINTER y QPGMR tienen un nivel de anotación de LOG(4 0 *NOLIST); por lo tanto, los mensajes y los textos de primer y segundo nivel para los mensajes se graban en las anotaciones de trabajo. No obstante, estas anotaciones no se imprimen a menos que especifique *LIST en el mandato SIGNOFF. Para cambiar el nivel de anotación de los trabajos interactivos, puede utilizar los mandatos CHGJOB o CHGJOBD.

Si un usuario de la estación de pantalla utiliza un menú suministrado por IBM o la pantalla de entrada de mandatos, se visualizan todos los mensajes de error. Si el usuario de la estación de pantalla utiliza un programa inicial escrito por el usuario, cualquier mensaje no supervisado hace que el programa inicial finalice y se produzca una anotación de trabajo. Sin embargo, si el programa inicial supervisa los mensajes, recibe el control cuando se recibe un mensaje. En este caso, puede ser importante asegurarse de que se produzcan anotaciones de trabajo para poder determinar el error específico que se produjo. Por ejemplo, supongamos que el programa inicial visualiza un menú que incluye una opción de desconexión, que toma el valor por omisión de *NOLIST. El programa inicial supervisa todas las excepciones e incluye el mandato Cambiar Variable (CHGVAR) que cambia la opción de desconexión por *LIST si se produce una excepción:

```
           PGM
           DCLF MENU
           DCL &SIGNOFFOPT TYPE(*CHAR) LEN(7) VALUE(*NOLIST)
           .
           .
           .
           MONMSG MSG(CPF0000) EXEC(GOTO ERROR)
  PROMPT:  SNDRCVF RCDFMT(PROMPT)
           CHGVAR &IN41 '0'
           .
           .
           .
           IF (&OPTION *EQ '90') SIGNOFF LOG(&SIGNOFFOPT)
           .
           .
           .
           GOTO PROMPT
  ERROR:   CHGVAR &SIGNOFFOPT '*LIST'
           CHGVAR &IN41 '1'
           GOTO PROMPT
           ENDPGM
```

Si se produce una excepción en el ejemplo anterior, el mandato CHGVAR cambia la opción del mandato SIGNOFF por *LIST y activa un indicador. Este indicador puede utilizarse para condicionar una constante que visualiza un mensaje que informa al usuario de la estación de pantalla que se ha producido un acontecimiento inesperado y le indica lo que debe hacer.

Si el trabajo interactivo está ejecutando un programa o procedimiento CL, los mandatos CL se anotan sólo si el nivel de anotación es 3 ó 4 y se cumple una de las siguientes condiciones:

- Se ha especificado LOG(*YES) en el mandato Crear Programa en Lenguaje de Control (CRTCLPGM).

- Se ha especificado LOG(*JOB) en el mandato Crear Programa en Lenguaje de Control (CRTCLPGM) y (*YES) es el atributo de trabajo LOGCLPGM actual.

  El atributo de trabajo LOGCLPGM puede establecerse y cambiarse utilizando el parámetro LOGCLPGM en los mandatos SBMJOB, CRTJOBD, CRTJOBD y CHGJOBD.

8.7.1.7 Consideraciones para Anotaciones de Trabajo por Lotes

Para sus aplicaciones por lotes, es posible que desee modificar la cantidad de información ya anotada. El nivel de anotación (LOG(4 0 *NOLIST)) especificado en la descripción de trabajo para el subsistema QBATCH suministrado por IBM proporciona una anotación completa si el trabajo finaliza de manera anómala. Si el trabajo finaliza con normalidad, no se producen anotaciones de trabajo.

Si desea imprimir las anotaciones de trabajo en todos los casos, utilice el mandato Cambiar Descripción de Trabajo (CHGJOBD) para cambiar la descripción de trabajo o especifique un valor LOG distinto en el mandato BCHJOB o SBMJOB. Remítase al apartado "Anotaciones de trabajo" en el tema 8.7.1 para ver una descripción de los niveles de anotación.

Si un trabajo por lotes está ejecutando un programa o procedimiento CL, los mandatos CL *siempre* se anotan si se especifica LOG(*YES) al crear módulos o programas utilizando los siguientes mandatos:

- Crear Programa de Lenguaje de Control (CRTCLPGM)
- Crear Módulo de Lenguaje de Control (CRTCLMOD)
- Crear Lenguaje de Control Enlazado (CRTBNDCL)

Los mandatos CL también se anotan si especifica LOGCLPGM(*YES) al utilizar los mandatos CHGJOB y SBMJOB.

8.7.2 Anotaciones Históricas QHST

Las anotaciones históricas (QHST) están formadas por una cola de mensajes y un archivo físico llamado versión de anotación. El sistema graba los mensajes enviados a la cola de mensajes de las anotaciones históricas en el archivo físico de la versión de anotación actual.

- Anotaciones históricas (QHST). Contiene un rastreo de alto nivel de las actividades del sistema, como puedan ser la información del sistema, del subsistema o de los trabajos, el estado de los dispositivo y los mensajes del operador del sistema. Su cola de mensajes es QHST.

Cuando una versión de anotación está llena, se crea automáticamente una nueva versión de las anotaciones. Cada versión es un archivo físico cuyo nombre sigue el siguiente formato:

```
Qxxxaadddn
```

donde:

xxx Es una descripción de 3 caracteres del tipo de anotación (HST)

aaddd Es la fecha en calendario Juliano del primer mensaje en la versión de la anotación

n Es un número de secuencia en la fecha de calendario Juliano (de A a Z, y de 0 a 9)

Nota: El número de registros en cada versión de las anotaciones históricas está especificado en el valor del sistema QHSTLOGSIZ.

El texto del archivo de anotaciones históricas contiene la fecha y hora del primer y último mensaje en la versión de anotaciones históricas. La fecha y la hora del primer mensaje se encuentran en las posiciones 1-13 del texto; la fecha y la hora del último mensaje en las posiciones 14-26. La fecha y la hora están en el formato cyymmddhhmmss, donde:

c Es el dígito del siglo

aammdd Es la fecha en que se envió el mensaje

hhmmss Es la hora en que se envió el mensaje

Puede crear un máximo de 36 versiones de anotaciones históricas con la misma fecha en calendario Juliano. Si se crean en el mismo día más de 36 versiones, el siguiente día disponible en calendario Juliano se utiliza para las subsiguientes versiones de anotaciones históricas. Si alguna de las antiguas versiones se suprimen, es posible utilizar los nombres de nuevo. Las versiones de anotaciones históricas secuenciadas por nombre, se encuentran fuera de orden si se suprimen versiones y los nombres se utilizan de nuevo.

También puede escribir un programa para procesar registros de anotaciones históricas. Puesto que puede haber disponibles varias versiones de las mismas anotaciones, debe seleccionar la versión de anotación que se va a procesar. Para determinar cuáles son las versiones de anotación disponibles, utilice el mandato Visualizar Descripción de Objeto (DSPOBJD). Por ejemplo, el siguiente mandato DSPOBJD visualiza las versiones de las anotaciones históricas que están disponibles:

```
DSPOBJD   OBJ(QSYS/QHST*)   OBJTYPE(*FILE)
```

Puede suprimir anotaciones del sistema utilizando la opción de supresión desde la pantalla que aparece con el mandato Trabajar con Objetos (WRKOBJ). En el mandato DLTLOG de QUSRTOOL encontrará un método automático para borrar anotaciones QHST.

Puede visualizar o imprimir la información de las anotaciones utilizando el mandato Visualizar Anotaciones (DSPLOG). Puede seleccionar la información que quiere visualizar o imprimir especificando una combinación de los siguientes elementos:

- Período de tiempo
- Nombre del trabajo que envió la entrada de las anotaciones
- Identificadores de mensajes de entradas

El siguiente mandato DSPLOG visualiza todas las entradas disponibles para el trabajo OEDAILY en el día actual:

```
DSPLOG JOB(OEDAILY)
```

La pantalla resultante es:

```
+-------------------------------------------------------------------------------+
|                                                                               |
|                    Visualizar Contenido de Anotaciones Históricas             |
|                                                                               |
|  Trabajo OEDAILY arrancado                                                    |
|  El archivo de base de datos OEMSTR en biblioteca OELIB ha caducado           |
|  Trabajo OEDAILY ha finalizado de forma anómala                               |
|  Trabajo OEDAILY arrancado                                                    |
|  Trabajo OEDAILY finalizado                                                   |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                    Final      |
| Pulse Intro para continuar.                                                   |
|                                                                               |
| F3=Salir  F10=Visualizar todo F12=Cancelar                                    |
|                                                                               |
|                                                                               |
+-------------------------------------------------------------------------------+
```

|Si restablece la fecha o la hora del sistema con un valor anterior, o la
|avanza en más de 48 horas, se arranca una nueva versión de anotaciones
|históricas. Ello asegura que todos los mensajes de una versión de
|anotaciones históricas individual, se encuentran en orden cronológico.

|Las versiones de anotaciones creadas en un release anterior a V3R6M0,
|deben contener entradas que no están en orden cronológico si la fecha y
|hora del sistema fue restablecido por un valor anterior. Por
|consiguiente, cuando intente visualizar la versión de anotación, deberán
faltar algunas entradas. Por ejemplo, si la versión de las anotaciones
contiene entradas fechadas en 1988 seguidas por entradas fechas en 1987 y
se desea visualizar las entradas de 1987, se especifican las fechas de
1987 en el parámetro PERIOD del mandato DSPLOG pero las entradas que
espera no se visualizan. Hay que utilizar siempre la fecha del sistema
(QDATE) y la hora del sistema (QTIME) o bien especificar el parámetro
PERIOD de la forma siguiente:

```
  PERIOD((hora-arranque fecha-arranque) (*AVAIL *END))
```

El sistema graba los mensajes enviados a una cola de mensajes de
anotaciones en el archivo físico de versión actual cuando la cola de
mensajes está llena o cuando se ha utilizado el mandato DSPLOG. Si desea
asegurarse de que la versión actual está actualizada, especifique un
identificador ficticio de mensajes, tal como **###0000**, en el mandato
DSPLOG. No se visualizan mensajes, pero se actualiza el archivo físico de
versión de anotación.

Si imprime la información de unas anotaciones utilizando el mandato
Visualizar Anotaciones y el parámetro de salida *PRINT, (DSPLOG
OUTPUT(*PRINT), sólo se imprime una línea de cada mensaje, utilizando los
105 primeros caracteres de cada mensaje.

Si imprime la información de unas anotaciones utilizando el mandato
Visualizar Anotaciones y el parámetro de salida *PRTWRAP, (DSPLOG
OUTPUT(*PRTWRAP), los mensajes de más de 105 caracteres se acomodan en
líneas adicionales hasta un máximo de 2000 caracteres.

Si visualiza la información de las anotaciones utilizando el mandato
Visualizar Anotaciones (DSPLOG), sólo se muestran 105 caracteres de texto
del mensaje. Los caracteres que sobrepasen los 105 se truncan por la
derecha.

8.7.3 Formato de las Anotaciones Históricas

Se utiliza un archivo de base de datos para almacenar los mensajes enviados a una anotación del sistema. Como todos los registros de un archivo físico tienen la misma longitud y, en cambio, no sucede lo mismo con los mensajes que se envían a las anotaciones, los mensajes pueden ocupar más de un registro. Cada uno de estos registros tiene tres campos:

- Fecha y hora del sistema (campo de caracteres de longitud 8). Éste es un campo interno. La fecha y hora convertidas también figuran en el mensaje.
- Número de registro (campo de 2 bytes). Por ejemplo, el campo contiene el valor hexadecimal **0001** para el primer registro, hexadecimal **0002** para el segundo, y así sucesivamente.
- Datos (campo de caracteres de longitud 132).

El tercer campo (datos) del primer registro tiene el siguiente formato:

Contenido	**Tipo**	**Longitud**	**Posiciones en registro**
Nombre del trabajo	Carácter	26	11-36
Fecha y hora convertidas(1)	Carácter	13	37-49
ID de mensaje	Carácter	7	50-56
Nombre de archivo de mensajes	Carácter	10	57-66
Nombre de biblioteca	Carácter	10	67-76
Tipo de mensaje(2)	Carácter	2	77-78
Código de gravedad	Carácter	2	79-80
Nombre del programa emisor(3)	Carácter	12	81-92
Número de instrucción de programa emisor(4)	Carácter	4	93-96
Nombre de programa receptor(3)	Carácter	10	97-106
Número de instrucción de programa receptor(4)	Carácter	4	107-110
Longitud del texto de mensaje	Binario	2	111-112
Longitud de los datos del mensaje	Binario	2	113-114
Identificador de juego de caracteres (CCSID) para texto o datos(5)	Binario	4	115-118
Reservado	Carácter	20	119-142

(1) El formato es: **caammddhhmmss**
siendo:

c El dígito del siglo (c=0 si aa >= 40, c = 1 si aa < 40)

aammdd El año, mes y día que se envía el mensaje

hhmmss La hora, minuto y segundo que se envía el mensaje

(2) Tiene el mismo valor que el parámetro RTNTYPE del mandato Recibir Mensaje (RCVMSG).

(3) Si el emisor o el receptor es un procedimiento ILE, la entrada de las anotaciones históricas sólo contiene el nombre del programa ILE. El nombre de módulo y el nombre de procedimiento no se incluyen en las anotaciones históricas.

(4) Si el emisor o el receptor es un procedimiento ILE, el número de instrucción de envío o recepción es 0.

(5) Este CCSID se aplica sólo a los datos de mensaje definidos como datos *CCHAR si el mensaje es un mensaje almacenado. Los demás datos del mensaje pueden considerarse 65.535. De lo contrario, éste es el CCSID del mensaje.

El tercer campo (datos) de los restantes registros tiene el formato siguiente:

Contenido	**Tipo**	**Longitud**
Mensaje	Carácter	Variable(1)
Datos de mensaje	Carácter	Variable(2)
(1) Esta longitud se especifica en el primer registro (posiciones 111 y 112) y no puede exceder de 132.		
(2) Esta longitud se especifica en el primer registro (posiciones 113 y 114).		

Un mensaje nunca se divide cuando arranca una nueva versión de anotación. El primer y el último registro de un mensaje siempre se encuentran en la misma versión de QHST.

Para obtener una descripción de los datos de un mensaje, véase el apartado "Definición de Variables de Sustitución" en el tema 7.2.4.

8.7.4 Proceso del Archivo QHST

Si utiliza un programa HLL para procesar el archivo QHST, tenga en cuenta que la longitud del mensaje puede variar en cada aparición del mensaje. Puesto que el mensaje contiene variables que se pueden sustituir, la longitud real del mensaje varía; por lo tanto, los datos del mensaje comienzan en una posición variable cada vez que se utiliza un mismo mensaje.

8.7.5 Mensajes de Arranque y Terminación de Trabajos QHST

El sistema da un formato especial a los mensajes de arranque y terminación de trabajos. En el caso de los mensajes CPF1124 (arranque de trabajo) y CPF1164 (terminación de trabajo), los datos de mensaje siempre comienzan en la posición 11 del tercer registro.

La contabilidad del trabajo proporciona más información que CPF1124 y CPF1164. Para funciones de contabilidad de trabajo sencillas, utilice el mensaje CPF1164.

Si está interesado en procesar la información de QHST, utilice el mandato CVTQHST en QUSRTOOL, que crea un archivo de longitud fija descrito externamente para la información de QHST.

En el mensaje CPF1164, la información del rendimiento no se visualiza como texto. Puesto que el mensaje se encuentra en las anotaciones QHST, los usuarios pueden escribir programas de aplicación para recuperar estos datos. Esta información del rendimiento tiene el formato siguiente:

La información de rendimiento se pasa como un valor de texto de sustitución de longitud variable. Esto significa que los datos están en una estructura que tiene como primera entrada la longitud de los datos. El tamaño del campo de longitud no está incluido en la longitud. Los primeros campos de datos de la estructura son las horas y las fechas en que el trabajo entró en el sistema y en que arrancó el primer paso de direccionamiento para el trabajo. Las horas están en el formato 'hh:mm:ss'. Los separadores siempre son dos puntos. Las fechas están en el formato definido en el valor del sistema QDATFMT y los separadores están en el valor del sistema QDATSEP. La hora y fecha en que el trabajo entró en el sistema preceden a la hora y fecha de arranque del trabajo en la estructura.

La hora y la fecha en que el trabajo entró en el sistema corresponden al momento en que el sistema se da cuenta de que hay que arrancar un trabajo (se reserva una estructura de trabajo para este trabajo). Para un trabajo interactivo, la hora de entrada del trabajo es la hora en que el sistema reconoce la contraseña. Para un trabajo por lotes, es la hora en que se procesa el mandato BCHJOB o SBMJOB. Para un trabajo de supervisión, ya sea lector o transcriptor, es la hora en que se procesa el mandato de arranque correspondiente y para los trabajos de autoarranque es durante el arranque del subsistema.

A continuación de las horas y fechas está el tiempo de respuesta total y el número de transacciones. El tiempo total de respuesta está expresado en segundos y contiene el valor acumulado de todos los intervalos que el trabajo estuvo procesándose desde que se pulsó la tecla Intro en la estación de trabajo hasta que muestra la pantalla siguiente. Esta información es similar a la mostrada en la pantalla WRKACTJOB. Este campo sólo tiene sentido para los trabajo interactivos.

En el caso de una anomalía del sistema o de una finalización anómala del trabajo, también es posible que la última transacción no se incluya en el total. El código de finalización del trabajo en este caso sería de 40 o superior. La cuenta de transacciones también tiene significado sólo para los trabajos interactivos que no sean trabajos de consola y es el número de intervalos de tiempos de respuesta contados por el sistema durante el trabajo.

El número de operaciones de E/S auxiliares síncronas sigue al número de transacciones. Es lo mismo que el campo AUXIO que aparece en la pantalla WRKACTJOB con la excepción que este valor es el total del trabajo. Si finaliza el trabajo con un código final de 70, es posible que este valor no contenga la cuenta para el paso de direccionamiento final. Además, si existe un trabajo durante una IPL (utilizando un mandato TFRBCHJOB) y finaliza antes de estar activo después de una IPL, el valor es 0.

El campo final en las estadísticas de rendimiento es el tipo de trabajo. Los valores para este campo son:

A Trabajo arrancado automáticamente

B Trabajo por lotes

I Trabajo interactivo

M Supervisor del subsistema

R Lector de spool

S Trabajo del sistema

W Transcriptor de spool

X Arrancar trabajo

En el caso de los mensajes en los que los datos comienzan en una posición variable, puede acceder a los datos del mensaje haciendo lo siguiente:

- Determine la longitud de las variables del mensaje. Por ejemplo, supongamos que un mensaje utiliza las cinco variables siguientes:

  ```
  Nombre del trabajo *CHAR 10
  Nombre de usuario  *CHAR 10
  Número de trabajo  *CHAR 6
  Hora               *CHAR 8
  Fecha              *CHAR 8
  ```

  Estas variables aparecen siempre en las primeras 42 posiciones de los datos del mensaje.

- Para encontrar la ubicación de los datos del mensaje, tenga en cuenta que:

  - El mensaje comienza siempre en la posición 11 del segundo registro.

  - La longitud del mensaje se almacena en un campo de 2 posiciones que comienza en la posición 111 del primer registro. Esta longitud se almacena en un valor binario de modo que si la longitud del mensaje es de 60, el campo contiene hex **003C**.

  A continuación, utilizando la longitud y la posición de inicio del mensaje, se puede determinar la ubicación de los datos del mensaje.

Subtemas
8.7.5.1 Ejemplo de Proceso del Archivo QHST

8.7.5.1 Ejemplo de Proceso del Archivo QHST

|Puede utilizar un programa RPG para AS/400 que procesa el archivo QHST para diferentes mensajes y para el mensaje de terminación de trabajo CPF1164. Consulte el miembro PRTQHSTANL del archivo QATTINFO en la biblioteca QUSRTOOL para obtener documentación y ejemplos de fuente para procesar archivos QHST.

8.7.6 Supresión de Archivos QHST

Los archivos físicos de versión de anotación se acumulan en un sistema, y es necesario suprimir periódicamente las anotaciones antiguas que ya no se precisen. Las versiones de anotaciones se crean de tal modo que sólo las puede suprimir el responsable de seguridad.

La función Operational Assistant* proporciona una función de borrado que incluye la supresión de los archivos QHST antiguos. Otras alternativas son:

- Como encargado de seguridad, especificar:

  ```
  WRKOBJ  OBJ(QSYS/QHST*)  OBJTYPE(*FILE)
  ```

  Utilizar la opción 4 para borrar archivos viejos que ya no son necesarios.
- La herramienta DLTQHST en QUSRTOOL proporciona un método de proceso por lotes de borrado de archivos QHST antiguos. Vea el miembro DLTQHST del archivo QATTINFO en la biblioteca USRTOOL para obtener documentación sobre esta herramienta.

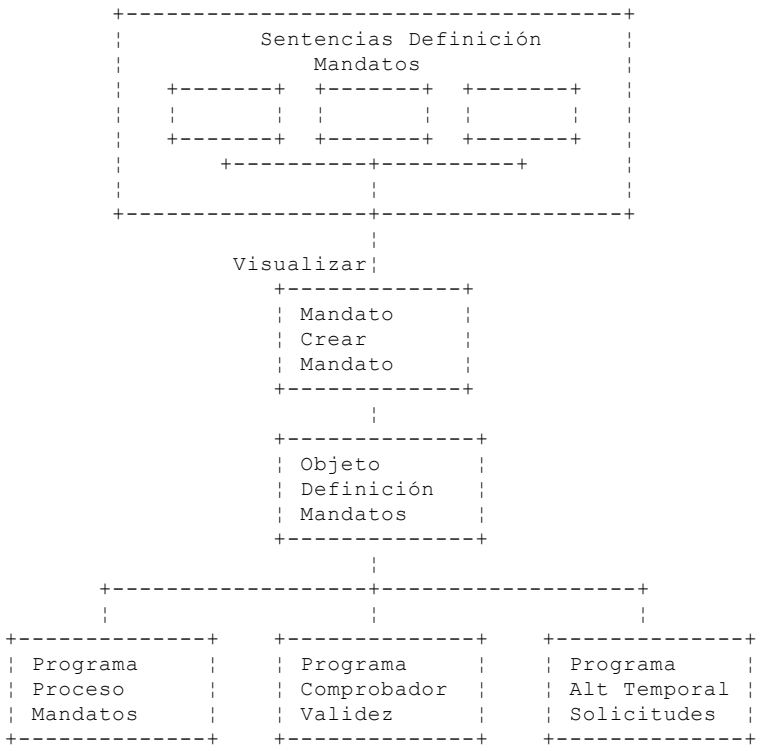
9.0 Capítulo 9. Definición de Mandatos

Un mandato CL es una sentencia que solicita que el sistema realice una función. Esta función la efectúa un programa que se ejecuta cuando se entra el mandato. Los mandatos CL le permiten solicitar una amplia gama de funciones. Puede utilizarlos tal como se los facilita IBM, cambiar sus valores por omisión o definir sus propios mandatos. En este capítulo se describe cómo puede definir y crear sus propios mandatos.

Subtemas

9.1 Visión General de Cómo Definir Mandatos

En la siguiente ilustración se muestran los pasos necesarios para crear un mandato. El texto que sigue a la ilustración describe cada paso.



La creación de sus propios programas de comprobación de validez y de alteración temporal de solicitudes son pasos opcionales.

Subtemas

9.1.1 Sentencias de Definición de Mandatos

Las sentencias de definición de mandatos contienen la información que se precisa para solicitar entrada al usuario de la estación de trabajo, dar por válida dicha entrada y definir los valores que se pasarán al programa que se llama cuando se ejecuta el mandato.

Puede haber sentencias de definición de mandatos en cualquier archivo soportado como entrada del mandato CRTCMD. Por ejemplo, los archivos fuente de SEU (Programa de Utilidad para Entrada del Fuente), los archivos en disquete y otros archivos de dispositivo pueden contener sentencias de definición de mandatos. Normalmente, el SEU las entra en un archivo fuente. La Tabla 9-1 contiene las sentencias utilizadas para definir mandatos.

Tabla 9-1. Sentencias para definir mandatos CL

Tipo de sentencia	**Nombre de sentencia**	**Descripción**
Mandato	CMD	Especifica el texto de solicitud, si existe, para el nombre de mandato
Parámetro	PARM	Define un parámetro o un parámetro clave para un mandato
Elemento	ELEM	Define un elemento de una lista utilizado como valor de parámetro
Calificador	QUAL	Define un calificador de un nombre utilizado como parámetro
Dependencia	DEP	Define la relación entre parámetros
Control de texto de solicitud	PMTCTL	Define las condiciones bajo las que se solicitan ciertos parámetros.

9.1.2 Mandato Crear Mandato (CRTCMD)

El mandato CRTCMD procesa las sentencias de definición de mandatos con el fin de crear el objeto de definición de mandato. El mandato CRTCMD puede ejecutarse interactivamente o en un trabajo de proceso por lotes.

9.1.3 Objeto de Definición de Mandato

El objeto de definición de mandato es el objeto que comprueba un programa del sistema para asegurarse de que el mandato es válido y que se han entrado los parámetros adecuados.

9.1.4 Comprobación de Validez

El sistema lleva a cabo una comprobación de validez de los mandatos. También puede escribir sus propios programas de comprobación de validez, si bien ello no resulta necesario.

La comprobación de validez que efectúa el sistema garantiza que:

- Se entran valores para los parámetros requeridos.
- Todos los valores de parámetro cumplen los requisitos de tipo de datos y longitud.
- Todos los valores de parámetro cumplen los requisitos opcionales especificados en la definición de mandato de:
  - Una lista de valores válidos
  - Un rango de valores
  - Una comparación relacional con un valor
- No se entran parámetros que se excluyen mutuamente.

El sistema realiza comprobación de validez cuando:

- Se entran mandatos interactivamente desde una estación de pantalla.
- Se entran mandatos desde una corriente de entrada por lotes que utiliza spool.
- Se entran mandatos en un archivo de base de datos mediante el Programa de Utilidad para Entrada del Fuente (SEU).
- Una llamada desde un HLL pasa un mandato al programa QCMDEXC, QCMDCHK o QCAPCMD. Véase el Capítulo 6 para obtener más información sobre el programa QCMDEXC.
- Se crea un módulo CL o un programa OPM.
- Los mandatos se ejecutan por un procedimiento o programa CL o un procedimiento REXX
- Un mandato se ejecuta utilizando la función de sistema de lenguaje C.

Si necesita más comprobación de validez que la que realiza el sistema, puede escribir un programa llamado *programa de comprobación de validez* (véase el apartado "Escritura de un Programa de Comprobación de Validez" en el tema 9.11) o puede incluir la comprobación en el programa de proceso de mandatos. Especifique los nombres de los programas de comprobación de validez y de proceso de mandato en el mandato CRTCMD.

Si se decide a escribir un programa de comprobación de validez, los valores de los parámetros de los mandatos se pasan al programa de comprobación de validez antes de pasar al programa de proceso de mandatos. Si todos los parámetros necesarios tienen especificadas unas constantes, también se llamará al programa que escriba durante la comprobación de la sintaxis de las corrientes de entrada en spool, así como cuando se utilice el SEU para entrar sus mandatos en un archivo de base de datos. Si se detecta un error, se emite un mensaje al usuario para que corrija todas las anomalías inmediatamente. El programa de proceso de mandatos puede suponer que los datos que se le pasan son correctos. Para que un programa de comprobación de validez envíe un mensaje del sistema, debe recibir el mensaje procedente de una cola de mensajes y colocarlo en la variable de sustitución &2 del mensaje CPD0006. El sistema operativo utiliza los primeros cuatro caracteres de la variable &1 del mensaje.

9.1.5 Programa de Alteración Temporal de Solicitudes

Pueden escribirse programas de alteración temporal de solicitudes para proporcionar valores actuales a los valores por omisión de los parámetros cuando se solicita el mandato. Para más detalles, véase el apartado "Utilización de Parámetros Clave y un Programa de Alteración Temporal de Solicitudes" en el tema 9.6. Los programas para alterar temporalmente una solicitud son opcionales.

9.1.6 Proporcionar Información de Ayuda para Mandatos

Para facilitar información de ayuda en línea sobre su mandato, puede utilizar grupos de paneles de ayuda. Un grupo de paneles es un objeto con tipo *PNLGRP. Para obtener más información acerca de los grupos de paneles de ayuda, consulte el manual Application Display Programming.

9.1.7 Programa de Proceso de Mandatos

El programa de proceso de mandatos (CPP) es aquel programa al cual llama el mandato para realizar la función solicitada. El CPP puede ser un programa CL, REXX o HLL. Por ejemplo, puede ser un programa de aplicación al que llama el mandato, o puede ser un programa CL o procedimiento REXX que contenga un mandato del sistema o una serie de mandatos.

El CPP debe aceptar los parámetros tal como los definen las sentencias de definición de mandatos.

9.1.8 Autorización Necesaria para los Mandatos que se Definen

Para que los usuarios utilicen un mandato que Ud haya creado, deben poseer autorización de operación para los mandatos y autorización sobre datos para el programa de proceso de mandatos y para el programa opcional de comprobación de validez. También deben tener autorización de lectura para la biblioteca que contiene el mandato, para el programa de proceso de mandatos y para el programa de comprobación de validez. Si se ejecutan otros mandatos o si se abren archivos, el usuario también ha de poseer la autorización adecuada para estos archivos o programas de proceso de mandatos.

9.1.9 Ejemplo de Creación de un Mandato

Si desea crear un mandato para que el operador del sistema llame a un programa con el fin de iniciar el sistema, haría lo siguiente (en este ejemplo se supone que está utilizando archivos fuente proporcionados por IBM):

1. Entre la sentencia fuente de definición del mandato en el archivo fuente QCMDSRC utilizando el nombre del miembro de STARTUP.

```
CMD PROMPT('Mandato S para STARTUP')
```

2. Cree el mandato entrando el mandato siguiente.

```
CRTCMD CMD(S) PGM(STARTUP) SRCMBR(STARTUP)
```

3. Entre las sentencias fuente para el programa STARTUP (programa de proceso de mandatos).

```
PGM
STRSBS QINTER
STRSBS QBATCH
STRSBS QSPL
STRPRTWTR DEV(QSYSPRT) OUTQ(QPRINT) WTR(WTR)
STRPRTWTR DEV(WSPR2) OUTQ(WSPRINT) WTR(WTR2)
SNDPGMMSG MSG('Procedimiento STARTUP finalizado') MSGTYPE(*COMP)
ENDPGM
```

4. Cree el programa utilizando el mandato Crear Programa CL Enlazado (CRTBNDCL).

```
CRTBNDCL STARTUP
```

En el ejemplo anterior, S es el nombre del nuevo mandato (especificado por el parámetro CMD). STARTUP es el nombre del programa de proceso de mandatos (especificado por el parámetro PGM) y también el nombre del miembro fuente que contiene la sentencia de definición del mandato (especificado por el parámetro SRCMBR). Ahora el operador del sistema puede entrar **S** para llamar al mandato o **CALL STARTUP** para llamar al programa de proceso de mandatos.

9.2 Cómo Definir Mandatos

Para crear un mandato, en primer lugar es necesario definirlo mediante sentencias de definición de mandatos, descritas detalladamente en la publicación CL Reference. A continuación se muestra el formato general de las sentencias de definición de mandatos y un resumen de las reglas de codificación.

Sentencia	**Reglas de codificación**
CMD	Sólo debe utilizarse una única sentencia CMD, que puede colocarse en cualquier parte del archivo fuente.
PARM	Se permite un máximo de 75 sentencias PARM. El orden en que las entre en el archivo fuente determina el orden en que deben especificarse los parámetros a la hora de procesar. Se requiere una sentencia PARM para cada parámetro que haya que pasar al programa de proceso de mandatos. Para especificar un parámetro como parámetro clave, debe especificar KEYPARM(*YES) para la sentencia PARM. El número de parámetros codificados con KEYPARM(*YES) no debe sobrepasar la cantidad necesaria para definir de forma exclusiva el objeto que se va a modificar. Para utilizar parámetros clave, debe especificarse el programa de alteración temporal de solicitudes al crear el mandato. No se permiten parámetros clave con el control de solicitud.
ELEM	En una lista se permiten 300 sentencias ELEM como máximo. El orden en que las entre en el archivo fuente determina el orden que tendrán los elementos en la lista. La primera sentencia ELEM debe tener una etiqueta de sentencia que coincida con la etiqueta de sentencia del parámetro TYPE de la sentencia PARM o ELEM de la lista.
QUAL	Se permiten 300 calificadores como máximo para un nombre calificado. El orden en que entre las sentencias QUAL en el archivo fuente determina el orden en que deben especificarse los calificadores y el orden en que se pasan al programa de comprobación de validez y al programa de proceso de mandatos.
DEP	La sentencia DEP debe estar situada después de todas las sentencias PARM a las que hace referencia. Por lo tanto, suelen aparecer al final del archivo fuente.
PMTCTL	La sentencia PMTCTL debe estar situada después de todas las sentencias PARM a las que hace referencia. Por lo tanto, suelen aparecer al final del archivo fuente.

Las sentencias ELEM o QUAL del archivo fuente deben ir precedidas de una sentencia PARM como mínimo. El mandato CRTCMD utiliza el archivo fuente en el que entra las sentencias de definición de mandatos cuando crea el mandato. Para obtener información referente a cómo entrar sentencias en un archivo fuente, consulte la publicación *Screen Design Aid User's Guide and Reference*, SC09-1340 o la publicación *DB2/400 Programación de la base de datos*.

Subtemas

9.2.1 Utilización de la Sentencia CMD

Cuando defina un mandato, debe incluir una única sentencia CMD con las sentencias de definición de mandatos.

Cuando define un mandato, puede proporcionar texto de solicitud para el usuario. Si éste elige que se le solicite el mandato en lugar de entrarlo por completo, el usuario escribe el nombre del mandato y pulsa F4 (Solicitud). Se visualizará la solicitud del mandato con el nombre del mandato y el texto de solicitud de cabecera en la línea 1 de la pantalla.

Si desea especificar el texto de solicitud para el mandato, utilice el parámetro PROMPT en la sentencia CMD para especificar la cabecera de la solicitud. Después especifique las solicitudes para los parámetros, los elementos de una lista y los calificadores en los parámetros PROMPT de las sentencias PARM, ELEM y QUAL.

En el parámetro PROMPT de la sentencia CMD, puede especificar el texto de cabecera de solicitud real como una serie de caracteres de 30 caracteres como máximo, o bien puede especificar el identificador de mensaje de una descripción de mensaje. en el ejemplo siguiente, se especifica una serie de caracteres para el mandato ORDENTRY.

```
  CMD PROMPT('Entrada de pedidos')
```

La línea 1 de la solicitud aparece como ésta después de que el usuario teclee el nombre del mandato y pulse F4.

```
  Entrada de pedidos (ORDENTRY)
```

Si no facilita texto de solicitud para el mandato que está definiendo, sólo es necesario utilizar la palabra CMD para la sentencia CMD. Sin embargo, tal vez desee utilizar la palabra clave PROMPT con fines de documentación.

9.2.2 Definición de Parámetros

Puede definir hasta 75 parámetros para cada mandato. Para definir un parámetro, debe utilizar la sentencia PARM.

En la sentencia PARM, especifique lo siguiente:

- El nombre de la palabra clave del parámetro
- Si el parámetro es o no un parámetro clave
- El tipo de valor de parámetro que puede pasarse
- La longitud del valor
- Si es necesario, el valor por omisión para el parámetro.

Además, hay que tener en cuenta la información siguiente al definir los parámetros. (El parámetro asociado a la sentencia PARM se pone entre paréntesis.)

- Si el programa de proceso de mandatos (RTNVAL) devuelve un valor. Si se especifica RTNVAL (*YES), debe codificarse una variable de retorno en el mandato cuando se le llama si desea que se devuelva el valor. Si no se especifica ninguna variable para un parámetro RTNVAL(*YES), se pasa un puntero nulo al programa de proceso de mandatos.

- Si el parámetro no va a aparecer en la solicitud al usuario, sino que se va a pasar al programa de proceso de mandatos como constante (CONSTANT).

- Si el parámetro está restringido (RSTD) a valores específicos (indicados en el parámetro VALUES, SPCVAL o SNGVAL) o si pueden incluir cualquier valor que coincida con el tipo de parámetro, longitud, rango de valor y una relación especificada.

- Cuáles son los valores específicos de parámetro válidos (VALUES, SPCVAL y SNGVAL).

- Qué pruebas deben realizarse con el valor del parámetro para determinar su validez (REL y RANGE).

- Si el parámetro es opcional o necesario (MIN).

- Cuántos valores se pueden especificar para un parámetros que precisa una lista sencilla (MIN y MAX).

- Si pueden entrarse datos no imprimibles (cualquier carácter con un valor hexadecimal de **00** a **3F** o **FF**) para el valor de parámetro (ALWUNPRT).

- Si puede entrarse un nombre de variable para el valor del parámetro (ALWVAR).

- Si el valor es un nombre de programa (PGM).

- Si el valor es un nombre de área de datos (DTAARA).

- Si el valor es un nombre de archivo (FILE).

- Si el valor debe tener la longitud exacta especificada (FULL).

- Si debe darse la longitud del valor con el valor (VARY).

- Si pueden especificarse expresiones para un valor de parámetro (EXPR).

- Si debe facilitarse información de atributos sobre el valor pasado para el parámetro (PASSATR).

- Si se pasa un valor al programa de proceso de mandatos o al programa de comprobación de validez si no se especifica el parámetro que se está definiendo (PASSVAL).

- Si se preserva el valor de mayúsculas/minúsculas o si se convierte a mayúsculas (CASE).

- Si la lista en la lista de desplazamientos (LISTDSPL) son valores de 2 bytes o 4 bytes

- Cuál es el identificador de mensaje o el texto de solicitud para el parámetro (PROMPT).

- Qué valores válidos se muestran en los campos de selección posibles de la pantalla de solicitud (CHOICE).

- Si un programa proporciona los valores de selección (CHOICEPGM).

- Si la solicitud de un parámetro la controla otro parámetro (PMTCTL).

- Si los valores para una sentencia PMTCTL los facilita un programa (para parámetros a los que se hace referencia en palabras clave CTL) (PMTCTLPGM).

- Si el valor estará oculto en las anotaciones de trabajo o cuando el mandato está solicitándose (DSPINPUT).

Subtemas

9.2.2.1 Denominación de la Palabra Clave para el Parámetro

El nombre de la palabra clave que elige para un parámetro debe describir la información que se solicita en el valor del parámetro. Por ejemplo, USUAR para nombre de usuario, VALCOMP para valor de comparación y TIPOEP para el tipo de entrada de pedidos. La palabra clave puede tener una longitud de hasta 10 caracteres alfanuméricos, el primero de los cuales debe ser alfabético.

9.2.2.2 Tipos de Parámetros

Los tipos de parámetros básicos son (valor del parámetro TYPE entre paréntesis):

- Decimal (*DEC). El valor de parámetro es un número decimal, que se pasa al programa de proceso de mandatos como valor decimal empaquetado cuya longitud es la especificada en el parámetro LEN. Los valores especificados con más dígitos fraccionarios de los definidos para el parámetro se truncan.

- Lógico (*LGL). El valor de parámetro es un valor lógico, '1' ó '0', que se pasa al programa de proceso de mandatos como una serie de caracteres con una longitud de 1 (**F1** ó **F0**).

- Carácter (*CHAR). El valor del parámetro es una serie de caracteres, que puede incluirse entre apóstrofos y que se pasa al programa de proceso de mandatos como una serie de caracteres cuya longitud es la especificada en el parámetro LEN. El valor se pasa con los apóstrofos eliminados, justificado por la izquierda y rellenado con espacios en blanco.

- Nombre (*NAME). El valor de parámetro es una serie de caracteres que representa un nombre básico. La longitud máxima del nombre es de 256 caracteres; el primer carácter es alfabético (A-Z), $, # o @ Los demás caracteres son los mismos que el primer carácter pero también pueden incluir los números 0 a 9, subrayados (_) y puntos (.). El nombre también puede ser una serie de caracteres que empiece y termine por comillas dobles ("). El valor se pasa al programa de proceso de mandatos como serie de caracteres de una longitud especificada en el parámetro LEN. Este valor está justificado por la izquierda y se rellena con espacios en blanco. El tipo *NAME se utiliza normalmente para nombres de objetos. Si se puede entrar un valor especial, como *LIBL o *NONE, para el valor del parámetro nombre, debe describir el valor especial en el parámetro SPCVAL. Después, si el usuario de la estación de pantalla entra un valor especial para los valores del parámetro nombre, las normas para la verificación de nombre se pasan por alto.

- Nombre simple (*SNAME). El valor del parámetro es una serie de caracteres que sigue las mismas normas de denominación que *NAME, excepto en que no se permiten puntos (.).

- Nombre de comunicaciones (*CNAME). El valor del parámetro es una serie de caracteres que sigue las mismas normas de denominación que *NAME, excepto en que no se permiten puntos (.) o subrayados (_).

- Nombre de vía (*PNAME). El valor del parámetro es una serie de caracteres, que puede incluirse entre apóstrofos y que se pasa al programa de proceso de mandatos como una serie de caracteres cuya longitud es la especificada en el parámetro LEN. El valor se pasa con los apóstrofos eliminados, justificado por la izquierda y rellenado con espacios en blanco. Consulte el manual *Introducción al Sistema de archivos integrado (IFS)* para obtener información referente a las reglas de utilización del valor del parámetro *PNAME.

- Nombre genérico (*GENERIC). El valor del parámetro es un nombre genérico, que acaba con un asterisco (*). Si el nombre no termina con un asterisco, se supone que el nombre genérico es un nombre de objeto completo. Un nombre genérico identifica un grupo de objetos cuyos nombres empiezan todos con los caracteres que preceden al asterisco. Por ejemplo, INV* identifica los objetos cuyos nombres empiezan con INV, como INV, INVOICE e INVENTORY. El nombre genérico se pasa al programa de proceso de mandatos para que éste pueda encontrar los nombres de objeto que empiecen con los caracteres del nombre genérico.

- Fecha (*DATE). El valor del parámetro es una serie de caracteres que se pasa al programa de proceso de mandatos con el formato **saammdd** (**s** = dígito de siglo, **a** = año, **m** = mes, **d** = día). El sistema establece el dígito de siglo basándose en el año de la fecha especificada en el parámetro del mandato. El dígito de siglo es 0 si **aa** es un número del 40 al 99 y 1 si **aa** es un número del 00 al 39. El usuario debe entrar la fecha en el parámetro de la fecha del mandato, en el formato especificado por el atributp de trabajo de formato de fecha (DATFMT). El atributo de trabajo separador de la fecha (DATSEP) determina qué carácter separador opcional se utilizará cuando se entre la fecha. Los atributos de trabajo DATFMT y DATSEP pueden modificarse con el mandato Cambiar Trabajo (CHGJOB).

- Hora (*TIME). El valor del parámetro es una serie de caracteres que se pasa al programa de proceso de mandatos con el formato **hhmmss** (**h** = hora, **m** = minuto, **s** = segundo).

- Hexadecimal (*HEX). El valor del parámetro es un valor hexadecimal.

Los caracteres especificados deben estar comprendidos entre 0 y F. El valor se pasa al CPP como caracteres hexadecimales (EBCDIC, 2 dígitos hexadecimales por byte), está ajustado por la derecha y rellenado con ceros. Si el valor está entre apóstrofos, es necesario que la cantidad de dígitos sea par.

- Cero elementos (*ZEROELEM). El valor del parámetro se considera un lista con cero elementos para la cual no se puede especificar ningún valor en el mandato. Este tipo de parámetro se utiliza para evitar que se entre un valor para un parámetro que es una lista, aunque el programa de proceso de mandatos (CPP) espera un valor. Por ejemplo, si dos mandatos utilizan el mismo CPP, uno podría pasar una lista para un parámetro y el otro puede no tener valores para pasar. El parámetro para el segundo mandato se definiría con TYPE(*ZEROELEM).

- Entero (*INT2 ó *INT4). El valor del parámetro es un número entero pasado como un número binario con signo de 2 ó 4 bytes. Los programas CL no dan soporte a los valores binarios en las variables.

- Nulo (*NULL). El valor del parámetro es un puntero nulo, que siempre se pasa al programa de proceso de mandatos como un área de retención de posición. Las únicas palabras clave PARM válidas para este parámetro son KWD, MIN y MAX.

- Serie de mandatos (*CMDSTR). El valor del parámetro es un mandato. Se pasa al CPP como una serie de mandatos. Los parámetros del mandato del tipo *CMDSTR no permiten variables CL como entrada.

- Etiqueta de sentencia. Identifica la primera de una serie de sentencias QUAL o ELEM que describen más detalladamente el nombre calificado o la lista mixta definida por esta sentencia PARM.

Los tipos de parámetros siguientes sólo son para los mandatos proporcionados por IBM.

- Expresión (*X). El valor del parámetro es una serie de caracteres, un nombre de variable o un valor numérico. El valor se pasa como un valor numérico si contiene solamente dígitos, un signo más o menos y/o una coma decimal; de lo contrario, se pasa como una serie de caracteres.

- Nombre de variable (*VARNAME). El valor del parámetro es un nombre de variable que se pasa al programa de mandatos como una serie de caracteres. Este valor está justificado por la izquierda y se rellena con espacios en blanco. Una variable es un nombre que hace referencia a un valor de datos real durante el proceso. Un nombre de variable puede tener una longitud máxima de 10 caracteres alfanuméricos (el primero de los cuales debe ser alfabético) y debe ir precedido de un signo &; por ejemplo, &PARM. Si el nombre de una variable no sigue el convenio de denominación utilizado en el sistema AS/400, debe colocarlo entre apóstrofos.

- Mandato (*CMD). El valor del parámetro es un mandato. Por ejemplo, el mandato CL IF tiene un parámetro denominado THEN cuyo valor debe ser otro mandato.

9.2.2.3 Longitud del Valor de Parámetro

También puede especificar una longitud (parámetro LEN) para los valores de parámetro de los tipos que se indican a continuación. En el caso de los parámetros de fecha u hora, la fecha siempre tiene una longitud de 7 caracteres y la hora de 6 caracteres. A continuación se muestra la longitud máxima para cada tipo de parámetro y la longitud por omisión para cada tipo de parámetro para el que se pueda especificar una longitud.

Tipo de datos	**Longitud máxima**	**Longitud por omisión**
*DEC	24 (9 posiciones decimales)	15 (5 posiciones decimales)
*LGL	1	1
*CHAR	5000	32
*NAME	256	10
*SNAME	256	10
*CNAME	256	10
*GENERIC	256	10
*HEX	256	1
*X	(256 24 9)	(1 15 5)
*VARNAME	11	11
*CMDSTR	20K	256
*PNAME	5000	32

La longitud máxima mostrada aquí es la longitud máxima permitida para estos tipos de parámetros cuando se ejecuta el mandato. Sin embargo, la longitud máxima permitida para las constantes de tipo carácter en las sentencias de definición de mandatos es de 32 caracteres. Esta restricción es aplicable a los parámetros CONSTANT, DFT, VALUES, REL, RANGE, SPCVAL y SNGVAL. Hay longitudes específicas para los campos de entrada disponibles al solicitar un mandato CL. Las longitudes de los campos de entrada son de 1 a 12 caracteres y 17, 25, 32, 50, 80, 132, 256 y 512 caracteres. Si un parámetro determinado tiene una longitud distinta de las permitidas, el campo de entrada se visualiza con la siguiente longitud de campo superior.

9.2.2.4 Valores Por Omisión

Si define un parámetro opcional, puede definir un valor en el parámetro DFT para utilizarlo si el usuario no entra el valor del parámetro en el mandato. Este valor se denomina *valor por omisión*. El valor por omisión debe reunir todos los requisitos para aquel parámetro (tal es el caso del tipo, longitud y valores especiales). Si no especifica un valor por omisión para para un parámetro opcional, se utilizan los siguientes valores por omisión:

Tipo de datos	**Valor por omisión**
*DEC	0
*INT2	0
*INT4	0
*LGL	'0'
*CHAR	Blancos
*NAME	Blancos
*SNAME	Blancos
*CNAME	Blancos
*GENERIC	Blancos
*DATE	Ceros ('F0')
*TIME	Ceros ('F0')
*ZEROELEM	0
*HEX	Ceros ('00')
*NULL	Nulo
*CMDSTR	Blancos
*PNAME	Blancos

9.2.2.5 Ejemplo de Definición de un Parámetro

En el ejemplo siguiente se define un parámetro OETYPE para que un mandato llame a una aplicación de entradas de pedidos.

```
  PARM   KWD(OETYPE)  TYPE(*CHAR)  RSTD(*YES) +
         VALUES(DAILY WEEKLY MONTHLY)  MIN(1) +
         PROMPT('Tipo de entrada de pedidos:')
```

El parámetro OETYPE es necesario (el parámetro MIN es 1) y su valor está restringido (el parámetro RSTD es *YES) a los valores DAILY, WEEKLY o MONTHLY. El parámetro PROMPT contiene el texto de solicitud para el parámetro. Puesto que no se ha especificado la palabra clave LEN y se ha definido TYPE(*CHAR), la longitud por omisión es de 32.

9.3 Restricciones de Parámetros y Tipos de Datos

El siguiente cuadro muestra las combinaciones válidas de los parámetros según el tipo de parámetro. Una **X** indica que la combinación es válida; un número hace referencia a una restricción reflejada al final de la tabla.

	LEN	RTNVAL	CONSTANT	RSTD	DFT	VALUES	RANGE	SPCVAL	SNGVAL	MIN	MAX	ALWVAR	ALWUNPR	ALWPRT	DFTARR	FILE	FULL	EXPR	VARY	PASSTR	PASVAL	CASE	LISTDSP	LISTPUT	CHOICE
*DEC	X	4	X	X	X	X	X	X	6	1	X	X		X					X		5	X	9	X	X	X
*LGL	X	4	X	X	X	X			6	1	X	X		X					X	X	5	5	X	9	X	X	X
*CHAR	X	4	X	X	X	X	X	X	6	1	X	X	X	X	X	X	X	X	X	X	5	5	X	9	X	X	X
*NAME	X		X	X	X	X	X	X	6	1	X	X		X	X	X	X	X	X	X	5	5	X	9	X	X	X
*SNAME	X		X	X	X	X	X	X	6	1	X	X		X	X	X	X	X	X	X	5	5	X	9	X	X	X
*CNAME	X		X	X	X	X	X	X	6	1	X	X		X	X	X	X	X	X	X	5	5	X	9	X	X	X
*PNAME	X	4	X	X	X	X	X	X	6	1	X	X	X	X	X	X	X	X	X	X	5	5	X	9	X	X	X
*GENERIC	X		X	X	X	X	X	X	6	1	X	X		X	X	X	X	X	X	X	5	5	X	9	X	X	X
*DATE			X	X	X	X	X	X	6	1	X	X		X					X		5	X	9	X	X	X
*TIME			X	X	X	X	X	X	6	1	X	X		X					X		5	X	9	X	X	X
*HEX	X		X	X	X	X	X	X	6	1	X	X		X					X	X		5	X	9	X	X	X
*ZEROELE											X	X												9			
*INT2			X	X	X	X	X	X	6	1	X	X		X					X		5	X	9	X	X	X
*INT4			X	X	X	X	X	X	6	1	X	X		X					X		5	X	9	X	X	X
*CMDSTR	X		X		X					2	3		8							5	5		9	X	X	X
*NULL	X									2	3												9			
STMT LABEL			X		X				X	X	X			X	X	X					7	9			X

Notas:

1 Válido solamente si el valor de MAX es mayor que 1. Asimismo, se pasan por alto los valores destino cuando CPP es un procedimiento REXX. Los valores pasados como parámetros de procedimiento REXX son los valores tecleados o los valores por omisión para cada parámetro.
2 El valor de MIN no puede exceder de 1 para TYPE(*NULL).
3 El valor de MAX no puede exceder de 1 para TYPE(*NULL) o TYPE(*CMDSTR).
4 No es válido cuando el mandato de CPP es un procedimiento REXX.
5 Se pasa por alto el parámetro cuando el CPP es un procedimiento REXX.
6 Los valores destino se pasan por alto cuando el CPP es un procedimiento REXX. Los valores pasados como parámetros de procedimiento REXX son los valores tecleados o los valores por omisión para cada parámetro.
7 PASSVAL pasa una palabra clave sin espacios en blanco ni otros caracteres entre paréntesis cuando el CPP es un procedimiento REXX.
8 El valor ALWVAR se pasa por alto para este tipo de parámetros. Las variables CL no están permitidas cuando el tipo de parámetro es *CMDSTR.
9 El parámetro CASE (*MIXED) sólo se permite con el tipo *CHAR y *PNAME.

El cuadro que figura a continuación muestra las restricciones y combinaciones de parámetros válidas para las sentencias PARM, ELEM y QUAL. Por ejemplo, la intersección de la fila de LEN y la columna de DFT está en blanco; por lo tanto, no hay restricciones y la combinación de LEN(XX) y DFT(XX) es válida. Sin embargo, la intersección de la fila de DFT y la columna de CONSTANT contiene un **4**, que hace referencia a una nota situada al final de la tabla en la que se describe la restricción.

	LEN	RTNVAL	CONSTANT	RSTD	DFT	VALUES	REL	RANGE	SPCVAL	SNGVAL	MIN	MAX	ALWUNPRT	ALWVAR	PGM	DTAARA	FILE	FULL	EXPR	VARY	PASSATR	PASSVAL	CASE	LISTDSPL	CHOICE	CHOICEPG	PMTCTL	PMTCTLPG	PROMPT
LEN																													
RTNVAL		1	1	1	1	1	1	1	1	1		2		13	1	1	1	1	1	3	3	15							
CONSTANT		1			4					21		2							5					20	16				
RSTD		1				7	9	9	7	7																			
DFT		1	4								8																		
VALUES		1		7																									
REL		1		9				9																					
RANGE		1		9			9															12							
SPCVAL		1		7																									
SNGVAL		1	21	7								11																	
MIN					8						10											15							
MAX		2	2							11	10																		
ALWUNPRT																													
ALWVAR		13																											
PGM		1														14	14												
DTAARA		1													14		14												
FILE		1														14	14												
FULL		1																											
EXPR		1	5																										
VARY		3																		3									
PASSATR		3																			3								
PASSVAL		15						12			15																		
CASE																							22						
LISTDSPL																								23					
CHOICE			16																										
CHOICEPG																										17			
PMTCTL			18								19																		
PMTCTLPG			18																										
PROMPT			6																										

Notas:

1 El parámetro RTNVAL no puede utilizarse con ninguno de los parámetros siguientes: CONSTANT, RSTD, DFT, VALUES, REL, RANGE, SPCVAL, SNGVAL, PGM, DTAARA, FILE, FULL o EXPR. El parámetro RTNVAL no puede utilizarse en ningún mandato utilizando un procedimiento REXX como un CPP.

2 No está permitido un valor de MAX superior a 1.

3 Si se especifican PASSATR(*YES) y RTNVAL(*YES), también debe especificarse VARY(*YES). Si se especifican RTNVAL(*YES) y VARY(*YES), debe utilizar *INT2 o *INT4. Las combinaciones de *INT2 y *INT4 no son válidas.

4 Los parámetros CONSTANT y DFT se excluyen mutuamente.

5 Los parámetros EXPR(*YES) y CONSTANT se excluyen mutuamente.

6 No está permitido el parámetro PROMPT.

7 Si se especifica el parámetro RSTD, uno de los parámetros siguientes también debe especificarse: VALUES, SPCVAL o SNGVAL.
8 El valor de MIN debe ser 0.
9 Los parámetros REL, RANGE y RSTD(*YES) se excluyen mutuamente.
10 El valor especificado para el parámetro MIN no debe exceder el valor especificado para el parámetro MAX.
11 El valor MAX debe ser mayor que 1 y/o el tipo de parámetro debe ser una etiqueta de sentencia.
12 El parámetro no puede hacer referencia a un parámetro definido con el parámetro PASSVAL(*NULL). Un rango entre parámetros no es válido en una sentencia PARM definida con PASSVAL(*NULL).
13 Si se especifica RTNVAL(*YES), no puede especificarse ALWVAR(*NO).
14 PGM(*YES), DTAARA(*YES) y un valor distinto de *NO para los parámetros FILE se excluyen mutuamente.
15 PASSVAL(*NULL) no está permitido con RTNVAL(*YES) o un valor superior a 0 para MIN.
16 Los parámetros CHOICE y CONSTANT se excluyen mutuamente.
17 CHOICE(*PGM) necesita un nombre para CHOICEPGM.
18 CONSTANT es mutuamente excluyente con los parámetros PMTCTL y PMTCTLPGM.
19 PMTCTL no está permitido con un valor superior a 0 para MIN.
20 CONSTANT es mutuamente excluyente con DSPINPUT(*NO) y DSPINPUT(*PROMPT).
21 El parámetro CONSTANT no puede estar definido en la sentencia ELEM/QUAL si en la sentencia PARM hay definido un parámetro SNGVAL.
22 El parámetro CASE sólo es válido en las sentencias PARM y ELEM. CASE no es válido en la sentencia QUAL.
23 El parámetro LISTDSPL sólo es válido en la sentencia PARM.

9.4 Definición de Listas para Parámetros

Puede definir un parámetro para aceptar una lista de valores en lugar de un único valor. Puede definir los siguientes tipos de lista:

- Una lista sencilla, que permite especificar uno o más valores del mismo tipo para un parámetro
- Una lista mixta, que permite especificar un conjunto de valores definidos por separado para un parámetro
- Una lista dentro de una lista, que permite especificar una lista más de una vez para un parámetro o que permite especificar una lista para un valor en una lista mixta

El siguiente ejemplo de mandato fuente ilustra los distintos tipos de lista existentes:

```
                CMD        PROMPT('Ejemplo de mandato de listas')

     /* EL PARÁMETRO SIGUIENTE ES UNA LISTA SENCILLA. ACEPTARÁ HASTA 5   */
     /* NOMBRES.                                                        */
                PARM       KWD(SIMPLST) TYPE(*NAME) LEN(10) DFT(*ALL) +
                             SPCVAL((*ALL)) MAX(5) PROMPT('Lista sencilla +
                             de hasta 5 nombres')

     /* EL PARÁMETRO SIGUIENTE ES UNA LISTA MIXTA DE 3 VALORES, CADA UNO */
     /* DE DISTINTO TIPO Y/O LONGITUD. LOS ELEMENTOS NO PUEDE REPETIRSE. */
                PARM       KWD(MXDLST) TYPE(MLSPEC) PROMPT('Lista +
                         mixta de 3 val')
  MLSPEC:     ELEM       TYPE(*CHAR) LEN(4) PROMPT('Elem 1 de 3')
              ELEM       TYPE(*DEC) LEN(3 0) PROMPT('Segundo de tres')
              ELEM       TYPE(*CHAR) LEN(10) PROMPT('Último de tres +
                           elementos')

  /* EL PARÁMETRO SIGUIENTE ES UNA LISTA DENTRO DE UNA LISTA. CONTIENE    */
  /* UNA LISTA DE HASTA 2 ELEMENTOS, QUE PUEDEN REPETIRSE HASTA 3 VECES.  */
                PARM       KWD(LWITHINL1) TYPE(LWLSPECA) MAX(3) +
                             PROMPT('Lista repetible de 2 elementos')
  LWLSPECA:     ELEM         TYPE(*CHAR) LEN(10) PROMPT('Parte 1 de +
                               lista repetible')
                ELEM         TYPE(*DEC) LEN(5 0) PROMPT('Parte 2 de +
                               lista repetible')

  /* EL PARÁMETRO SIGUIENTE ES UNA LISTA DENTRO DE UNA LISTA. CONTIENE    */
  /* LISTA DE HASTA 2 ELEMENTOS, EL PRIMERO DE LOS CUALES PUEDE          */
  /* REPETIRSE HASTA 3 VECES.                                            */
              PARM       KWD(LWITHINL2) TYPE(LWLSPECB) MAX(1) +
                           PROMPT('Repetible sencilla en una mixta')
  LWLSPECB:   ELEM       TYPE(*CHAR) LEN(10) MAX(3) PROMPT('Lista  +
                           sencilla en una lista')
              ELEM       TYPE(*DEC) LEN(5 0) PROMPT('Parám. único +
                           en una lista')
```

La pantalla siguiente muestra la solicitud para el ejemplo de mandato anterior:

```
+-------------------------------------------------------------------------------+
|                                                                               |
|                         Ejemplo de mandato de listas (LSTEXAMPLE)             |
|                                                                               |
|  Teclee elección, pulse Intro.                                                |
|                                                                               |
|  Lista sencilla de hasta 5 nombres . . SIMPLST        *ALL                    |
|                            + para más valores                                 |
|  Lista mixta de 3 valores          MXDLST                                     |
|    Elem 1 de 3  . . . . . . . . .                                             |
|    Segundo de tres  . . . . . . .                                             |
|    Último de tres elementos . . .                                             |
|  Lista repetible de 2 elementos    LWITHINL1                                  |
|    Parte 1 de lista repetible . .                                             |
|    Parte 2 de lista repetible . .                                             |
|                            + para más valores                                 |
|  Repetible sencilla en una mixta   LWITHINL2                                  |
|    Lista sencilla en una lista  .                                             |
|                            + para más valores                                 |
|    Parám. único en una lista  . .                                             |
|                                                                               |
|                                                                               |
|                                                                       Final   |
|  F3=Salir  F4=Lista  F5=Renovar   F12=Cancelar  F13=Ayuda del solicitante     |
```

```
| F24=Más teclas                                                                 |
|                                                                                |
|                                                                                |
+--------------------------------------------------------------------------------+
```

Subtemas

9.4.1 Definición de una Lista Sencilla

Una lista sencilla puede aceptar uno o varios valores del tipo especificado por el parámetro. Por ejemplo, si el parámetro corresponde al nombre de usuario, una lista sencilla indica que se puede especificar más de un nombre de usuario en ese parámetro.

```
  USER(JONES SMITH MILLER)
```

Si el valor de un parámetro es una lista sencilla, especifique el número máximo de elementos que la lista puede aceptar utilizando el parámetro MAX en la sentencia PARM. Para una lista sencilla, no es necesario especificar más sentencias de definición de mandatos que la sentencia PARM.

El ejemplo siguiente define un parámetro USER para el que el usuario de estación de pantalla puede especificar hasta cinco nombres de usuario (lista sencilla).

```
  PARM     KWD(USER) TYPE(*NAME) LEN(10) MIN(0) MAX(5) +
           SPCVAL(*ALL) DFT(*ALL)
```

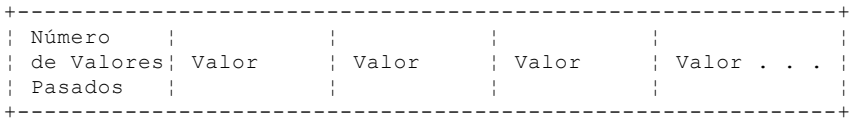
Éste es un parámetro opcional, tal como lo especifica MIN(0), y el valor por omisión es *ALL, tal como indica DFT(*ALL).

Cuando los elementos de una lista sencilla se pasan al programa de proceso de mandatos, el formato varía según se esté utilizando CL, HLL o REXX. En el apartado siguiente se describe cómo se pasan los elementos utilizados en el ejemplo anterior utilizando CL y HLL. Para ver una explicación de las diferencias al utilizar REXX, remítase al apartado "Utilización de REXX para Listas Sencillas" en el tema 9.4.1.2.

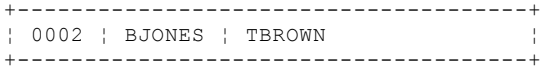
Subtemas

9.4.1.1 Utilización de CL o HLL para Listas Sencillas

cuando el mandato se ejecuta utilizando CL o HLL, los elementos de una lista sencilla se pasan al programa de proceso de mandatos con el formato siguiente:



El número de valores pasados se especifica mediante un valor binario de dos caracteres de longitud. Este número indica cuántos valores pueden entrarse (cuántos se pasan) realmente, no cuántos se pueden especificar. Los valores se pasan por tipo de parámetro, igual que en el caso de un valor de un único parámetro (tal como se describe en el apartado 9.2.2). Por ejemplo, si se especifican dos nombres de usuario (BJONES y TBROWN) para el parámetro USER, se pasa lo siguiente.



Los nombres de usuario se pasan como valores de 10 caracteres ajustados por la izquierda y rellenados con espacios en blanco.

Cuando se pasa una lista sencilla, sólo se pasa el número de elementos especificado en el mandato. El almacenamiento inmediatamente después del último elemento que se pasa no forma parte de la lista y no debe hacerse referencia a él como parte de la misma. Por lo tanto, cuando un programa de proceso de mandatos (CPP) procesa una lista sencilla, utiliza el número de elementos pasados para determinar cuántos elementos se pueden procesar.

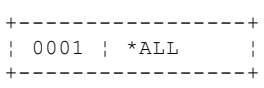
La Figura 9-1 muestra un ejemplo de un procedimiento CL que utiliza la función incorporada binaria para procesar una lista sencilla.

```
PGM PARM (...&USER...)
.
.
.
/* Declarar espacio para una lista sencilla de    */
/* hasta cinco valores de 10 caracteres a recibir */
DCL VAR(&USER) TYPE(*CHAR) LEN(52)
.
DCL VAR(&CT)    TYPE(*DEC)  LEN(3 0)
DCL VAR(&USER1) TYPE(*CHAR) LEN(10)
DCL VAR(&USER2) TYPE(*CHAR) LEN(10)
DCL VAR(&USER3) TYPE(*CHAR) LEN(10)
DCL VAR(&USER4) TYPE(*CHAR) LEN(10)
DCL VAR(&USER5) TYPE(*CHAR) LEN(10)
.
.
.
CHGVAR  VAR(&CT) VALUE(%BINARY(&USER 1 2))
.
IF (&CT > 0) THEN(CHGVAR &USER1 %SST(&USER 3 10))
IF (&CT > 1) THEN(CHGVAR &USER2 %SST(&USER 13 10))
IF (&CT > 2) THEN(CHGVAR &USER3 %SST(&USER 23 10))
IF (&CT > 3) THEN(CHGVAR &USER4 %SST(&USER 33 10))
IF (&CT > 4) THEN(CHGVAR &USER5 %SST(&USER 43 10))
IF (&CT > 5) THEN(DO)
/*  Si CT es mayor que 5, los valores pasados son  */
/*  mayores de lo que el programa espera y debe    */
/*  realizarse una lógica de error                 */
.
.
.
ENDDO
ELSE DO
/* Se pasa la cantidad correcta de valores */
/* y el programa puede seguir el proceso   */
.
.
.
ENDDO
ENDPGM
```

Figura 9-1. Ejemplo de lista sencilla

Se puede utilizar esta misma técnica para procesar otras listas en un procedimiento o programa CL.

Para una lista sencilla, puede entrarse en el mandato un valor único como *ALL o *NONE en lugar de la lista. Los valores únicos se pasan como un valor individual. De la misma forma, si no se especifican valores para un parámetro, el valor por omisión, si hay alguno definido, se pasa como el único valor de la lista. Por ejemplo, si el valor por omisión *ALL se utiliza para el parámetro USER, se pasa lo siguiente:



Se pasa *ALL como un valor de 10 caracteres ajustado por la izquierda y rellenado con espacios en blanco.

Si no hay definido ningún valor por omisión para un parámetro opcional de lista sencilla, se pasa lo siguiente:

9.4.1.2 Utilización de REXX para Listas Sencillas

Cuando se ejecuta el mandato, los elementos de una lista sencilla se pasan al procedimiento REXX en la serie de argumentos con el formato siguiente:

```
. . . USER(valor1 valor2  . . . valorN) . . .
```

siendo **valorN** el último valor de la lista sencilla.

Por ejemplo, si se especifican dos nombres de usuario (BJONES y TBROWN) para el parámetro USER, se pasa lo siguiente.

```
. . . USER(BJONES TBROWN) . . .
```

Cuando se pasa una lista sencilla, sólo se pasa el número de elementos especificado en el mandato. Por lo tanto, cuando el CPP procesa una lista sencilla, utiliza el número de elementos que se han pasado para determinar cuántos elementos se pueden procesar.

El ejemplo de REXX de la Figura 9-2 produce el mismo resultados que el procedimiento CL de la Figura 9-1 en el tema 9.4.1.1:

```
.
.
.
PARSE ARG . 'USER(' user ')' .
.
.
CT = WORDS(user)
IF CT > 0 THEN user1 = WORD(user,1) else user1 = ''
IF CT > 1 THEN user2 = WORD(user,2) else user2 = ''
IF CT > 2 THEN user3 = WORD(user,3) else user3 = ''
IF CT > 3 THEN user4 = WORD(user,4) else user4 = ''
IF CT > 4 THEN user5 = WORD(user,5) else user5 = ''
IF CT > 5 THEN
  DO
   /*  Si CT es mayor que 5, los valores pasados son
       mayores de lo que el programa espera y debe
       realizarse una lógica de error                */
.
.
.
  END
ELSE
  DO
    /* Se pasa la cantidad correcta de valores
       y el programa puede seguir el proceso  */
  END
EXIT
```

Figura 9-2. Ejemplo de lista sencilla con REXX

Este mismo procedimiento puede utilizarse para procesar otras listas en un programa REXX.

Para una lista sencilla, puede entrarse en el mandato un valor único como *ALL o *NONE en lugar de la lista. Los valores únicos se pasan como un valor individual. De la misma forma, si no se especifican valores para un parámetro, el valor por omisión, si hay alguno definido, se pasa como el único valor de la lista. Por ejemplo, si el valor por omisión *ALL se utiliza para el parámetro USER, se pasa lo siguiente:

```
. . . USER(*ALL) . . .
```

Si no hay definido ningún valor por omisión para un parámetro opcional de lista sencilla, se pasa lo siguiente:

```
. . . USER() . . .
```

Para obtener más información acerca de los procedimientos REXX, consulte las publicaciones *REXX/400 Programmer's Guide*, SC24-5665 y *REXX/400 Reference*, SC24-5664.

9.4.2 Definición de una Lista Mixta

Una lista mixta acepta un conjunto de valores definidos por separado que generalmente tienen significados diferentes, son de tipos distintos y ocupan una posición fija en la lista. Por ejemplo, **LOG(4 0 *SECLVL)** podría especificar una lista mixta. El primer valor, **4**, identifica el nivel del mensaje que se ha de anotar; el segundo valor, **0**, es la gravedad inferior del mensaje a notar. El tercer valor, ***SECLVL**, especifica la cantidad de información que hay que anotar (los mensajes de primer y segundo nivel). Si el valor de un parámetro es una lista mixta, los elementos de la lista deben definirse por separado utilizando una sentencia Elemento (ELEM) para cada uno.

El parámetro TYPE en la sentencia PARM asociada debe tener una etiqueta que haga referencia a la primera sentencia ELEM para la lista.

```
           PARM    KWD(LOG)    TYPE(LOGLST) ...

  LOGLST:  ELEM    TYPE(*INT2)      ...
           ELEM    TYPE(*INT2)      ...
           ELEM    TYPE(*CHAR)      LEN(7)
```

La primera sentencia ELEM es la única de estas sentencias que puede tener una etiqueta. Especifique las sentencias ELEM en el orden en que los elementos figuran en la lista.

Observe que, cuando el parámetro MAX tiene un valor mayor que 1 en la sentencia PARM y el parámetro TYPE hace referencia a sentencias ELEM, el parámetro que se define es una lista dentro de una lista.

Los parámetros que puede especificar en la sentencia ELEM son TYPE, LEN, CONSTANT, RSTD, DFT, VALUES, REL, RANGE, SPCVAL, SNGVAL, MIN, MAX, ALWUNPRT, ALWVAR, PGM, DTAARA, FILE, FULL, EXPR, VARY, PASSATR, CHOICE, CHOICEPGM y PROMPT.

En el ejemplo siguiente, se define un parámetro CMPVAL para el cual el usuario de la estación de pantalla puede especificar un valor de comparación y una posición inicial para la comparación (lista mixta).

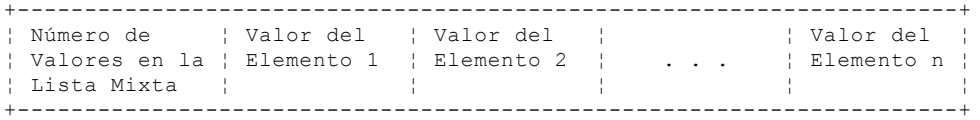
```
        PARM   KWD(CMPVAL) TYPE(CMP) SNGVAL(*ANY) DFT(*ANY) +
                MIN(0)
  CMP:  ELEM   TYPE(*CHAR) LEN(80) MIN(1)
        ELEM   TYPE(*DEC) LEN(2 0) RANGE(1 80) DFT(1)
```

Cuando los elementos de una lista mixta se pasan al programa de proceso de mandatos, el formato varía según esté utilizando CL, HLL o REXX. En el apartado siguiente se describe cómo se pasan los elementos utilizados en el ejemplo anterior utilizando CL y HLL. Para ver una explicación de las diferencias al utilizar REXX, remítase al apartado "Utilización de REXX para Listas Mixtas" en el tema 9.4.2.2.

Subtemas

9.4.2.1 Utilización de CL o HLL para Listas Mixtas

Cuando un mandato se ejecuta utilizando CL o HLL, los elementos de una lista mixta se pasan al programa de proceso de mandatos con el formato siguiente:



El número de valores que hay en la lista mixta se pasa como un valor binario de dos caracteres de longitud. Este valor siempre indica cuántos valores se han definido para la lista mixta, no cuántos se han entrado realmente en el mandato. Este valor puede ser 1 si el parámetro SNGVAL se entra o se pasa como valor por omisión. Si el usuario no entra un valor para un elemento, se pasa un valor por omisión. Los elementos se pasan de acuerdo con su tipo, al igual que ocurre con los valores de parámetros únicos (tal como se describe en el apartado "Definición de Parámetros" en el tema 9.2.2). Por ejemplo, si, en el ejemplo anterior, el usuario entra el valor de comparación QCMDI para el parámetro CMPVAL, pero no entra un valor para la posición inicial, cuyo valor por omisión es 1, se pasa lo siguiente:

Los datos QCMDI se pasan como un valor de 80 caracteres ajustado por la izquierda y rellenado con espacios en blanco. El número de elementos se envía como un valor binario de dos caracteres de longitud.

Cuando el usuario de la estación de pantalla entra un solo valor o cuando un único valor es el valor por omisión para una lista mixta, el valor se pasa como el primer elemento de la lista. Por ejemplo, si el usuario de la estación de pantalla entra *ANY como valor único para el parámetro, se pasa lo siguiente:

*ANY se pasa como un valor de 80 caracteres ajustado por la izquierda y rellenado con espacios en blanco.

Las listas mixtas se pueden procesar en programas CL. A diferencia de las listas sencillas, no es preciso comprobar el valor binario para determinar cuántos valores hay en la lista, ya que este valor es siempre el mismo para una lista mixta dada a menos que se haya pasado el parámetro SNGVAL al programa de proceso de mandatos. En este caso, el valor es 1. Si el mandato se entra con un único valor para el parámetro, sólo se pasaría dicho valor. Para procesar la lista mixta en un procedimiento CL, utilice la función incorporada de subserie (véase el Capítulo 2).

En un caso, sólo se pasa el valor binario 0000 como el número de valores para una lista mixta. Si no se define ningún valor por omisión en la sentencia PARM para un parámetro opcional y el primer valor de la lista es necesario (MIN(1)), el propio parámetro no es necesario; sin embargo, si se especifica algún elemento, el primer elemento es necesario. En este caso, si se entra el mandato sin especificar un valor para el parámetro, se pasa lo siguiente:

Un ejemplo de un parámetro así sería:

```
        PARM  KWD(KWD1)    TYPE(E1) MIN(0)
  E1:   ELEM  TYPE(*CHAR)  LEN(10)  MIN(1)
        ELEM  TYPE(*CHAR)  LEN(2)   MIN(0)
```

Si un procedimiento CL procesara este parámetro, el valor del parámetro se podría recibir en una variable CL de 14 caracteres. Los 2 primeros caracteres se podrían comparar con uno de los siguientes:

- una variable de 2 caracteres inicializada a hex 0000 utilizando la función %SUBSTRING.
- un 0 decimal utilizando la función incorporada %BINARY.

Un método estándar de procesar listas mixtas es la utilización del mandato EXTLST de QUSRTOOL.

9.4.2.2 Utilización de REXX para Listas Mixtas

Cuando se ejecuta un mandato utilizando REXX, los elementos de una lista mixta se pasan al programa de proceso de mandatos con el formato siguiente:

```
. . . CMPVAL(valor1 valor2  . . . valorN) . . .
```

siendo **valorN** el último valor de la lista mixta.

Si el usuario no entra un valor para un elemento, se pasa un valor por omisión. Por ejemplo, si, en el ejemplo anterior, el usuario entra el valor de comparación QCMDI para el parámetro CMPVAL, pero no entra un valor para la posición inicial, cuyo valor por omisión es 1, se pasa lo siguiente:

```
. . . CMPVAL(QCMDI 1)  . . .
```

Observe que los blancos de cola no se pasan con los valores REXX.

Cuando un usuario de la estación de pantalla entra un solo valor o cuando un único valor es el valor por omisión para una lista mixta, el valor se pasa como el primer elemento de la lista. Por ejemplo, si el usuario de la estación de pantalla entra *ANY como valor único para el parámetro, se pasa lo siguiente:

```
. . . CMPVAL(*ANY) . . .
```

Una vez más, observe que los blancos de cola no se pasan con los valores REXX.

Si no se define ningún valor por omisión en la sentencia PARM para un parámetro opcional y el primer valor de la lista es necesario (MIN(1)), el propio parámetro no es necesario. Sin embargo, si se especifica algún elemento, el primer elemento es necesario. En este caso, si se entra el mandato sin especificar un valor para el parámetro, se pasa lo siguiente:

```
. . . CMPVAL() . . .
```

9.4.3 Definición de Listas dentro de Listas

Una lista dentro de una lista puede ser:

- Una lista que se puede especificar más de una vez para un parámetro (lista sencilla o mixta)
- Una lista que se puede especificar para un valor dentro de una lista mixta

A continuación se facilita un ejemplo de listas dentro de una lista.

```
  STMT((START RESPND) (ADDDSP CONFRM))
```

El par externo de paréntesis contiene la lista que se puede especificar para el parámetro (la lista exterior), mientras que los pares de paréntesis interiores contiene una lista dentro de una lista (una lista interior).

En el ejemplo siguiente, se define una lista mixta con una lista sencilla. Se especifica una lista mixta y el valor MAX del parámetro PARM es mayor que 1; por lo tanto, la lista mixta se puede especificar como máximo el número de veces que se indica en el parámetro MAX.

```
          PARM KWD(PARM1) TYPE(LIST1) MAX(5)
  LIST1:  ELEM TYPE(*CHAR) LEN(10)
          ELEM TYPE(*DEC) LEN(3 0)
```

en este ejemplo, los dos elementos pueden especificarse hasta cinco veces. Cuando se entra un valor para este parámetro, puede aparecer de la forma siguiente:

```
  PARM1((VAL1 1.0) (VAR2 2.0) (VAR3 3.0))
```

En el ejemplo siguiente, se especifica una lista sencilla como valor en una lista mixta. En este ejemplo, el valor MAX de la sentencia ELEM es mayor que 1; por lo tanto, el elemento se puede repetir tantas veces como se haya especificado en el parámetro MAX.

```
          PARM KWD(PARM2) TYPE(LIST2)
  LIST2:  ELEM TYPE(*CHAR) LEN(10) MAX(5)
          ELEM TYPE(*DEC) LEN(3 0)
```

En este ejemplo, el primer elemento se puede especificar cinco veces como máximo, pero el segundo elemento sólo se puede especificar una vez. Cuando se entra un valor para este parámetro, puede aparecer de la forma siguiente:

```
  PARM2((NAME1 NAME2 NAME3) 123.0)
```

Cuando se pasan listas dentro de listas al programa de proceso de mandatos, el formato varía según se utilice CL, HLL o REXX. En el apartado siguiente se describe cómo se pasan elementos utilizando CL y HLL. Para ver una explicación de las diferencias al utilizar REXX, remítase al apartado "Utilización de REXX para Listas Dentro de Listas" en el tema 9.4.3.2.

Subtemas

9.4.3.1 Utilización de CL o HLL para Listas Dentro de Listas

Cuando un mandato se ejecuta utilizando CL o HLL, Una lista dentro de una lista se pasa al programa de proceso de mandatos con el formato siguiente:

El número de listas se pasa como un valor binario de dos caracteres de longitud. Después del número de listas, se pasa el desplazamiento para las listas (no los valores que se entraron en las listas). Cada desplazamiento se pasa como un valor binario de longitud 2 o de longitud 4, dependiendo del valor del parámetro LISTDSPL.

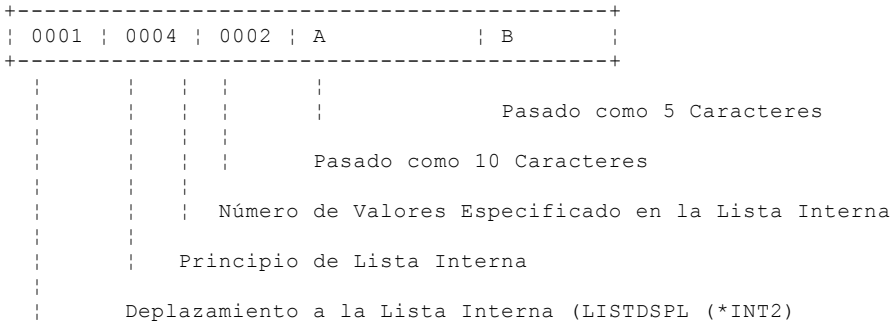
En el ejemplo siguiente se muestra una definición para un parámetro KWD2 (que es una lista mixta dentro de una lista sencilla), cómo puede el usuario de la estación de pantalla especificar el parámetro y qué se pasa. La definición de parámetro es:

```
         PARM    KWD(KWD2)    TYPE(LIST) MAX(20) MIN(0) +
                 DFT(*NONE)  SNGVAL(*NONE)  LISTDSPL(*INT2)
  LIST:  ELEM    TYPE(*CHAR)  LEN(10) MIN(1)       /*Desde valor*/
         ELEM    TYPE(*CHAR)  LEN(5) MIN(0)        /*Hasta valor*/
```

El usuario de la estación de pantalla entra el parámetro KWD2 así:

```
  KWD2((A B))
```

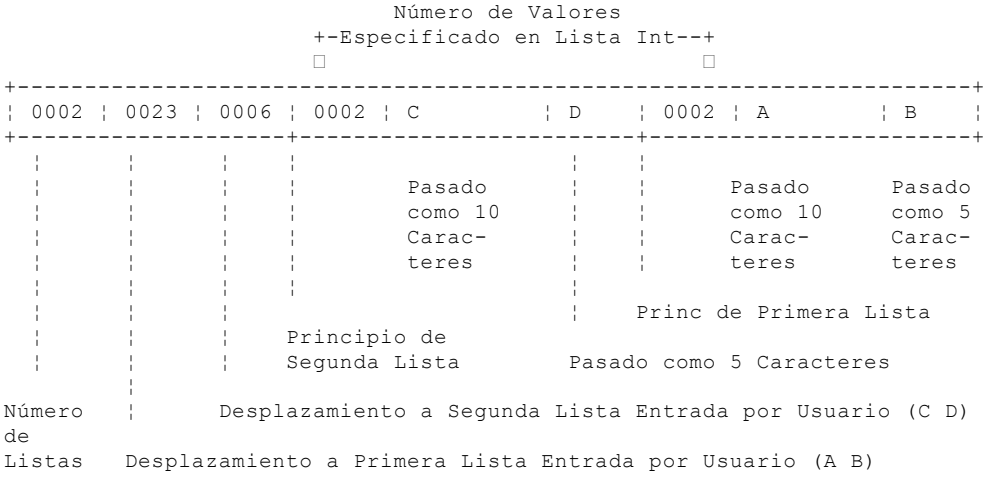
Se pasa lo siguiente al programa de proceso de mandatos:



En cambio, si el usuario de la estación de pantalla entra lo siguiente:

```
  KWD2((A B) (C D))
```

Se pasa lo siguiente al programa de proceso de mandatos:



Las listas dentro de una lista se pasan al programa de proceso de mandatos en el orden de n (la última introducida por el usuario de la estación de pantalla) hasta 1 (la primera introducida por el usuario de la estación de pantalla). Los desplazamientos, sin embargo, se pasan de 1 hasta n.

A continuación se facilita un ejemplo más complejo de listas dentro de listas. La definición de parámetro es:

```
          PARM    KWD(PARM1)  TYPE(LIST3)  MAX(25)
  LIST3:  ELEM    TYPE(LIST4)
          ELEM    TYPE(*CHAR)  LEN(3)
          ELEM    TYPE(*NAME)  LEN(2)  MAX(5)
  LIST4:  ELEM    TYPE(*DEC) LEN(7 2)
          ELEM    TYPE(*TIME)
```

Si el usuario de la estación de pantalla entra el parámetro PARM1 de este modo:

```
  PARM1(((11.1  120900)  A  (A1  A2))  ((-22.2  131500)  B  (B1  B2)))
```

Se pasa lo siguiente al programa de proceso de mandatos:

IMAGEN 10

9.4.3.2 Utilización de REXX para Listas Dentro de Listas

Cuando se ejecuta un mandato utilizando REXX, se pasa una lista dentro de una lista al programa de proceso de mandatos, de igual manera que se entran los valores para los parámetros. Los blancos de cola no se pasan.

El ejemplo siguiente muestra una definición para un parámetro KWD2, (que es una lista mixta dentro de una lista sencilla), cómo el usuario de la estación de pantalla puede especificar el parámetro y qué se pasa. La definición de parámetro es:

```
         PARM    KWD(KWD2)    TYPE(LIST) MAX(20) MIN(0) +
                 DFT(*NONE)  SNGVAL(*NONE)
  LIST:  ELEM    TYPE(*CHAR)  LEN(10) MIN(1)       /*Desde valor*/
         ELEM    TYPE(*CHAR)  LEN(5) MIN(0)        /*Hasta valor*/
```

El usuario de la estación de pantalla entra el parámetro KWD2 así:

```
  KWD2((A B))
```

Se pasa lo siguiente al programa de proceso de mandatos:

```
  KWD2(A B)
```

En cambio, si el usuario de la estación de pantalla entra lo siguiente:

```
  KWD2((A B) (C D))
```

Se pasa lo siguiente al programa de proceso de mandatos:

```
  KWD2((A B) (C D))
```

A continuación se facilita un ejemplo más complejo de listas dentro de listas. La definición de parámetro es:

```
          PARM    KWD(PARM1)  TYPE(LIST3)  MAX(25)
  LIST3:  ELEM    TYPE(LIST4)
          ELEM    TYPE(*CHAR)  LEN(3)
          ELEM    TYPE(*NAME)  LEN(2)  MAX(5)
  LIST4:  ELEM    TYPE(*DEC) LEN(7 2)
          ELEM    TYPE(*TIME)
```

El usuario de la estación de pantalla entra el parámetro PARM1 como:

```
  PARM1(((11.1  12D900)  A  (A1  A2))  ((-22.2  131500)  B  (B1  B2)))
```

Se pasa lo siguiente al programa de proceso de mandatos:

```
  PARM1(((11.1  12D900)  A  (A1  A2))  ((-22.2  131500)  B  (B1  B2)))
```

9.4.4 Definición de un Nombre Calificado

Un nombre calificado es el nombre de un objeto precedido del nombre de la biblioteca en la que dicho objeto está almacenado. Si un valor de parámetro o un elemento de lista es un nombre calificado, debe definir el nombre por separado utilizando sentencias Calificador (QUAL). Cada parte del nombre calificado debe estar definida en una sentencia QUAL. Las partes de un nombre calificado deben describirse en el orden en que aparecen el el nombre. Debe especificar *NAME o *GENERIC en la primera sentencia QUAL. La sentencia PARM o ELEM asociada debe identificar la etiqueta que hace referencia a la primera sentencia QUAL para el nombre calificado.

Las sentencias de definición de mandatos siguientes definen el nombre calificado más usual. Un objeto calificado está compuesto por el nombre de la biblioteca que contiene el objeto seguido por el nombre del propio objeto. Las sentencias QUAL deben aparecer en el orden en que aparecerán en el nombre calificado.

```
               PARM     KWD(NAME) TYPE(NAME1) SNGVAL(*NONE). . .

 +---------------------------------------+
 ¦
 +- NAME1:  QUAL     TYPE(*NAME)
            QUAL     TYPE(*NAME)
```

La mayor parte de los parámetros que se pueden especificar para la sentencia QUAL son los mismos que los descritos para la sentencia PARM (véase el apartado "Definición de Parámetros" en el tema 9.2.2). Sin embargo, para el parámetro TYPE sólo se pueden especificar los valores siguientes:

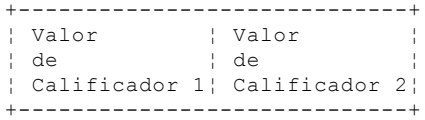
- *NAME
- *GENERIC
- *CHAR
- *INT2
- *INT4

Cuando se pasa un nombre calificado al programa de proceso de mandatos, el formato varía según se utilice CL, HLL o REXX. En el apartado siguiente se describe cómo se pasan nombres calificados utilizando CL y HLL. Para ver una explicación de las diferencias al utilizar REXX, remítase al apartado "Utilización de REXX para un Nombre Calificado" en el tema 9.4.4.2.

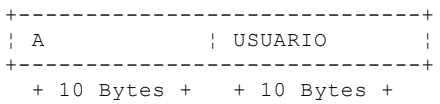
Subtemas
9.4.4.1 Utilización de CL o HLL para un Nombre Calificado
9.4.4.2 Utilización de REXX para un Nombre Calificado

9.4.4.1 Utilización de CL o HLL para un Nombre Calificado

Un nombre calificado se pasa al programa de proceso de mandatos con el formato siguiente al utilizar CL o HLL:

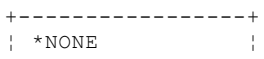


Por ejemplo, si el usuario de la estación de pantalla entra **NAME(USER/A)** para las sentencias QUAL definidas con anterioridad, el nombre se pasa al programa de proceso de mandatos de este modo:



Los calificadores se pasan al programa de proceso de mandatos de forma consecutiva por el tipo y la longitud, de igual modo que se pasan valores de parámetros únicos (tal como se describe en el apartado "Definición de Parámetros" en el tema 9.2.2). Los caracteres de separación (/) no se pasan. Esto es así independientemente de si se pasa un parámetro único, un elemento de una lista mixta o una lista sencilla de nombres calificados.

Si el usuario de la estación de pantalla entra un valor único para un nombre calificado, la longitud del valor pasado es el total de la longitud de las partes del nombre calificado. Por ejemplo, si define un nombre calificado con dos valores, cada uno de ellos de longitud 10, y si el usuario de la estación de pantalla entra un valor único, el valor único pasado se ajusta por la izquierda y se rellena por la derecha con espacios en blanco para que se pase un total de 20 caracteres. Si el usuario de la estación de pantalla entra *NONE como el valor único, se pasa el siguiente valor de 20 caracteres:



Los nombres calificados pueden procesarse en programas CL utilizando la función incorporada de subseries, tal y como se muestra en el siguiente ejemplo.

La función incorporada de subseries (%SUBSTRING o %SST) se utiliza para separar el nombre calificado en dos valores.

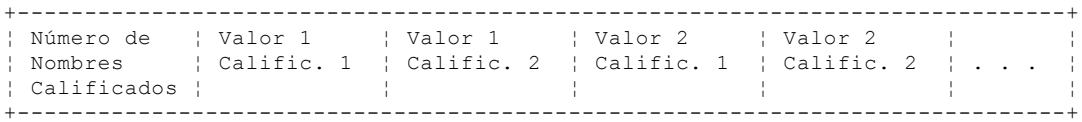
```
  PGM PARM(&QLFDNAM)
  DCL &QLFDNAM TYPE(*CHAR) LEN(20)
  DCL &OBJ TYPE(*CHAR) LEN(10)
  DCL &LIB TYPE(*CHAR) LEN(10)
  CHGVAR &OBJ %SUBSTRING(&QLFDNAM 1 10) /* Primer 10 */
  CHGVAR &LIB %SST(&QLFDNAM 11 10) /* Segundo 10 */
   .
   .
   .
  ENDPGM
```

Después puede especificar el nombre calificado con la sintaxis CL adecuada. Por ejemplo, **OBJ(&LIB/&OBJ)**.

También puede separarse el nombre calificado en dos valores utilizando el método siguiente:

```
  PGM PARM(&QLFDNAM)
  DCL &QLFDNAM TYPE(*CHAR) LEN(20)
  CHKOBJ (%SST(&QLFDNAM 11 10)/%SST(&QLFDNAM 1 10)) *PGM
   .
   .
   .
  ENDPGM
```

Se pasa una lista sencilla de nombres calificados al programa de proceso de mandatos en el formato siguiente:



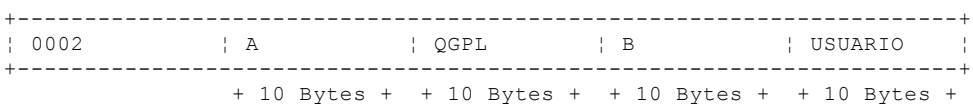
Por ejemplo, supongamos que se ha añadido MAX(3) de la forma siguiente a la sentencia PARM para el parámetro NAME.

```
          PARM   KWD(NAME) TYPE(NAME1) SNGVAL(*NONE) MAX(3)
  NAME1:  QUAL   TYPE(*NAME)
          QUAL   TYPE(*NAME)
```

Si el usuario de la estación de pantalla entrase lo siguiente:

```
  NAME(QGPL/A USER/B)
```

el parámetro de nombre se pasaría al programa de proceso de mandatos de este modo:



Si el usuario de la estación de pantalla entra el valor único NAME(*NONE), el parámetro de nombre se pasa como se indica a continuación:

9.4.4.2 Utilización de REXX para un Nombre Calificado

Cuando se ejecuta un mandato utilizando REXX, se pasa un nombre calificado al programa de proceso de mandatos del mismo modo que se entra el valor para el parámetro. Los blancos de cola no se pasan.

Por ejemplo, si un usuario de la estación de pantalla entra lo siguiente para las sentencias QUAL definidas anteriormente en este apartado:

```
  NAME(USER/A)
```

el nombre calificado se pasa al programa de proceso de mandatos con el formato siguiente:

```
  NAME(USER/A)
```

Los calificadores se pasan al programa de proceso de mandatos de forma consecutiva por el tipo y la longitud, de igual modo que se pasan valores de parámetros únicos (tal como se describe en el apartado "Definición de Parámetros" en el tema 9.2.2).

Si el usuario de la estación de pantalla entra *NONE como el valor único, se pasa el siguiente valor de 20 caracteres:

```
  NAME(*NONE)
```

El ejemplo siguiente muestra cómo un usuario de la estación de pantalla entraría una lista sencilla de nombres calificados:

```
  NAME(QGPL/A USER/B)
```

Utilizando REXX, el parámetro de nombre se pasaría al programa de proceso de mandatos de la forma siguiente:

```
  NAME(QGPL/A USER/B)
```

9.4.5 Definición de una Relación Dependiente

Si existe una relación necesaria entre parámetros y si los valores de parámetros deben comprobarse cuando se ejecuta el mandato, utilice la sentencia Dependiente (DEP) para definir esa relación. Con la sentencia DEP, puede:

- Especificar las condiciones de control que deben cumplirse antes de que las relaciones de parámetros definidas en el parámetro PARM tengan que ser verdaderas (CTL)
- Especificar las relaciones de parámetros que deben comprobarse si las condiciones de control definidas por CTL son verdaderas (PARM)
- Especificar el número de relaciones de parámetros definidas en la sentencia PARM asociada que deben ser verdaderas si la condición de control es verdadera (NBRTRUE)
- Especificar el identificador de un mensaje de error en un archivo de mensajes que ha de enviarse al usuario de la estación de pantalla si las dependencias de parámetros no se han cumplido

En el ejemplo siguiente, si el usuario de la estación de pantalla especifica el parámetro TYPE(LIST), debe especificar también el parámetro ELEMLIST.

```
  DEP CTL(&TYPE *EQ LIST) PARM(ELEMLIST)
```

En el ejemplo siguiente, el parámetro &WRITER no debe ser nunca igual al parámetro &NEWWTR. Si esta condición no es verdadera, se emite el mensaje **USR0001** al usuario de la estación de pantalla.

```
  DEP CTL(*ALWAYS) PARM((&WRITER *NE &NEWWTR)) MSGID(USR0001)
```

En el ejemplo siguiente, si el usuario de la estación de pantalla especifica el parámetro FILE, debe especificar también los parámetros VOL y LABEL.

```
  DEP CTL(FILE) PARM(VOL LABEL) NBRTRUE(*EQ 2)
```

9.4.6 Valores y Opciones Posibles

A la derecha del campo de entrada de las pantallas de solicitud, se visualizan solicitudes con las opciones posibles para los parámetros. El texto que ha de visualizarse puede crearse automáticamente, especificarse en el fuente de definición de mandatos o crearse dinámicamente mediante un programa de salida. Puede definirse texto que describa las opciones posibles para cualquier sentencia PARM, ELEM o QUAL, pero dadas las limitaciones del formato de pantalla, el texto se visualiza sólo para valores con una longitud de campo igual o inferior a 12, y no superior a 10 para todos los calificadores de un grupo excepto el primero.

El parámetro CHOICE define el texto para las opciones posibles. El valor por omisión para este parámetro es *VALUES, que indica que el texto va a crearse automáticamente a partir de los valores especificados para las palabras clave TYPE, RANGE, VALUES, SPCVAL y SNGVAL. El texto tiene como máximo 30 caracteres; si hay más valores que se ajustan a este tamaño, se añade una elipsis (...) al final del texto para indicar que está incompleto.

Puede especificar que no se visualice ninguna de las opciones posibles (*NONE) o bien que se visualice una serie de texto o el ID de un mensaje de texto que se recupera del archivo de mensajes especificado en el parámetro PMTFILE del mandato CRTCMD.

Asimismo, puede especificar que se ejecute un programa de salida durante la solicitud para proporcionar el texto de las opciones posibles. Esto puede hacerse si, por ejemplo, desea mostrar al usuario una lista de objetos que existen actualmente en el sistema. Puede utilizar el mismo programa de salida para proporcionar los valores mostrados en la pantalla Especificar Valor para Parámetro. Para especificar un programa de salida, especifique *PGM para el parámetro CHOICE y el nombre calificado del programa de salida en el parámetro CHOICEPGM de la sentencia PARM, ELEM o QUAL.

El programa de salida debe aceptar los dos parámetros siguientes:

- **Parámetro 1**: Un campo de 21 bytes pasado mediante una solicitud al programa de opciones que contiene lo siguiente:

  Posiciones Descripciones

  1-10 Nombre de mandato. Especifica el nombre del mandato que se procesa y que hace que el programa se ejecute.

  11-20 Nombre de la palabra clave. Especifica la palabra clave para la que se solicitan opciones posibles o valores permisibles.

  21 Los caracteres C o P indican que los datos se van a devolver. La letra C indica que se trata de un campo de 30 bytes en el que se devolverá el texto de las opciones posibles. La letra P indica que se trata de un campo de 2000 bytes en el que se devolverá una lista de valores permisibles.

- **Parámetro 2**: Un campo de 30 ó 2000 bytes para devolver alguna de las dos cosas siguientes:

  - Si C está en el byte 21 del primer parámetro (lo que indica que se devolverá el texto para las opciones posibles), se trata de un campo de 30 bytes en el que el programa colocará el texto.

  - Si P está en el byte 21 del primer parámetro (lo que indica que se va a devolver una lista de valores permisibles), se trata de un campo de 2000 bytes en el que el programa colocará la lista. Los primeros dos bytes de la lista deben contener el número de entradas (en binario) que hay en la lista. Este valor va seguido de entradas que constan de dos bytes binarios seguidos por el valor, que debe tener una longitud de 1 a 32 caracteres.

    Si se devuelve un valor de cero binario en los dos primeros bytes, no se visualizan valores permisibles.

    Si se devuelve un valor negativo binario en los dos primeros bytes, la lista de valores permisibles se toma del mandato.

Si se produce alguna excepción cuando se llama al programa, el texto de las opciones posibles se deja en blanco y la lista de valores permisibles se toma del mandato.

9.5 Utilización del Control de Solicitud

Puede controlar qué parámetros se visualizan para un mandato durante la solicitud utilizando las especificaciones de control de solicitud. Dicho control puede simplificar la solicitud para un mandato utilizando únicamente los parámetros que desea ver.

Puede especificar que se visualice un parámetro en función del valor especificado para otros parámetros. Esta especificación es útil cuando un parámetro tiene significado sólo cuando otro parámetro (denominado parámetro de control) tiene un cierto valor.

Puede también especificar que se seleccione un parámetro para la solicitud solamente si se solicitan los parámetros adicionales pulsando una tecla de función durante la solicitud. Esta especificación puede utilizarse para los parámetros que el usuario especifica raramente, bien sea porque normalmente se utiliza el valor por omisión o porque los parámetros controlan funciones poco utilizadas.

Si desea mostrar todos los parámetros de un mandato para el que se ha especificado el control de solicitud, puede solicitar que se visualicen todos los parámetros pulsando F9 durante la solicitud.

Subtemas

9.5.1 Solicitud Condicional

Cuando se solicita un mandato al usuario, se visualiza un parámetro condicionado por otros parámetros si:

- Está seleccionado por el valor especificado para el parámetro de control.
- El valor especificado para el parámetro de control es erróneo.
- Se especificó un valor para el parámetro condicionado.
- Durante la solicitud se pulsó una tecla de función para solicitar que se visualicen todos los parámetros.

Cuando se solicita al usuario un parámetro condicionado y no se ha especificado todavía ningún valor para su parámetro de control, se visualizan todos los parámetros seleccionados anteriormente. Cuando el usuario pulsa la tecla Intro, el parámetro de control se comprueba para determinar si el parámetro condicionado debe visualizarse o no.

Para especificar la solicitud condicional en el fuente de definición del mandato, especifique un nombre de etiqueta en el parámetro PMTCTL de la sentencia PARM para cada parámetro que esté condicionado por otro parámetro. La etiqueta especificada debe definirse en una sentencia PMTCTL que especifica el parámetro de control y la condición que está comprobándose para seleccionar el parámetro para la solicitud. Más de una sentencia PARM puede hacer referencia a la misma etiqueta.

En la sentencia PMTCTL, especifique el nombre del parámetro de control, una o más condiciones que han de comprobarse y el número de condiciones que deben ser válidas para seleccionar los parámetros condicionados para la solicitud. Si el parámetro de control tiene una correlación especial de valores, el valor entrado en la sentencia PMTCTL debe ser el valor destino. Si el parámetro de control es una lista o un nombre calificado, sólo se compara el primer calificador o elemento de la lista.

En el ejemplo siguiente, se seleccionarán los parámetros OUTFILE y OUTMBR solamente si se especifica *OUTFILE para el parámetro OUTPUT, y se seleccionará el parámetro OUTQ solamente si se especifica *PRINT para el parámetro OUTPUT.

```
       PARM OUTPUT TYPE(*CHAR) LEN(1) DFT(*) RSTD(*YES) +
              SPCVAL((*) (*PRINT P) (*OUTFILE F))
       PARM OUTFILE TYPE(Q1) PMTCTL(OUTFILE)
       PARM OUTMBR TYPE(*NAME) LEN(10) PMTCTL(OUTFILE)
       PARM OUTLINK TYPE(*CHAR) LEN(10)
       PARM OUTQ TYPE(Q1) PMTCTL(PRINT)
       Q1: QUAL TYPE(*NAME) LEN(10)
           QUAL TYPE(*NAME) LEN(10) SPCVAL(*LIBL) DFT(*LIBL)
  OUTFILE: PMTCTL CTL(OUTPUT) COND((*EQ F)) NBRTRUE(*EQ 1)
  PRINT:   PMTCTL CTL(OUTPUT) COND((*EQ P)) NBRTRUE(*EQ 1)
```

En este ejemplo, se solicita al usuario el parámetro OUTLINK después de haberse comprobado la condición para el parámetro OUTMBR. En ciertos casos, debe solicitarse al usuario el parámetro OUTLINK antes de que se haya comprobado el parámetro OUTMBR. Para especificar un orden de solicitud diferente, puede hacer dos cosas; ordene de nuevo los parámetros en el fuente de definición de mandatos o bien utilice la palabra clave PROMPT en la sentencia PARM para el parámetro OUTLINK.

Una etiqueta puede hacer referencia a un grupo de sentencias PMTCTL. Esto permite condicionar un parámetro con más de un parámetro de control. Para especificar un grupo de sentencias PMTCTL, entre la etiqueta en la primera sentencia del grupo. No pueden colocarse otras sentencias entre las sentencias PMTCTL del grupo.

Para especificar la relación lógica entre las sentencias del grupo utilice el parámetro LGLREL. Dicho parámetro no puede utilizarse en la primera sentencia PMTCTL de un grupo. Para las sentencias PMTCTL subsiguientes, el parámetro LGLREL especifica la relación lógica (*AND u *OR) con la(s) sentencia(s) PMTCTL que lo preceden. En cualquier combinación de relaciones *AND y *OR, las sentencias de un grupo pueden estar relacionadas lógicamente (primero se comprueban las relaciones *AND y después las *OR).

El siguiente ejemplo muestra cómo se utiliza la relación lógica para agrupar varias sentencias PMTCTL. En este ejemplo, el parámetro P3 se selecciona cuando se da cualquiera de las siguientes condiciones:

- Se especifica *ALL para P1.
- Se especifica *SOME para P1 y se especifica *ALL para P2.
- Se especifica *NONE para P1 y no se especifica *ALL para P2.

```
  PARM P1 TYPE(*CHAR) LEN(5) RSTD(*YES) VALUES(*ALL *SOME *NONE)
  PARM P2 TYPE(*NAME) LEN(10) SPCVAL(*ALL)
  PARM P3 TYPE(*CHAR) LEN(10) PMTCTL(PMTCTL1)
```

```
  PMTCTL1:PMTCTL CTL(P1) COND((*EQ *ALL))
          PMTCTL CTL(P1) COND((*EQ *SOME)) LGLREL(*OR)
          PMTCTL CTL(P2) COND((*EQ *ALL)) LGLREL(*AND)
          PMTCTL CTL(P1) COND((*EQ *NONE)) LGLREL(*OR)
          PMTCTL CTL(P2) COND((*NE *ALL)) LGLREL(*AND)
```

Puede especificarse un programa de salida para realizar un proceso adicional en un parámetro de control antes de que se compruebe. El programa de salida puede utilizarse para condicionar solicitudes según:

- El tipo u otro atributo de un objeto
- Un calificador o elemento de la lista con excepción del primero
- Una lista entera o nombre calificado

Para especificar un programa de salida, especifique el nombre calificado del programa en el parámetro PMTCTLPGM de la sentencia PARM para el parámetro de control. El programa de salida se ejecuta durante la solicitud, cuando se comprueba un parámetro. Las condiciones de la sentencia PMTCTL se comparan con el valor devuelto por el programa de salida, en vez de con el valor especificado para el parámetro de control.

Si el programa de salida no se encuentra o no se ejecuta satisfactoriamente, se coloca un solo asterisco en el valor devuelto. El programa de salida debe devolver un sólo asterisco cuando detecte un error. Tenga en cuenta estas consideraciones a la hora de codificar las sentencias PMTCTL; por ejemplo, visualizando todos los parámetros condicionados por ese parámetro cuando se produce un error.

El programa de salida debe aceptar tres parámetros:

- Un campo de 20 caracteres. La solicitud pasa el nombre del mandato en los diez primeros caracteres y el nombre del parámetro de control en los diez últimos. Este campo no debe modificarse.

- El valor del parámetro de control. Este campo tiene el mismo formato que cuando se pasa al programa de proceso de mandatos, y no debe modificarse.

- Un campo de 32 caracteres en el que el programa de salida coloca el valor que se ha de comprobar en las sentencias PMTCTL.

  El valor que se comprueba en la sentencia PMTCTL debe devolverse en el mismo formato que el tipo de datos declarados.

En el ejemplo siguiente, OBJ es un nombre calificado que puede corresponder a un mandato, programa o archivo. El programa de salida determina el tipo del objeto y devuelve el tipo en la variable &RTNVAL:

```
|  CMD:
   PARM OBJ TYPE(Q1) PMTCTLPGM(CNVTYPE)
   Q1: QUAL TYPE(*NAME) LEN(10)
       QUAL TYPE(*NAME) LEN(10) SPCVAL(*LIBL) DFT(*LIBL)
   PARM CMDPARM TYPE(*CHAR) LEN(10) PMTCTL(CMD)
   PARM PGMPARM TYPE(*CHAR) LEN(10) PMTCTL(PGM)
   PARM FILEPARM TYPE(*CHAR) LEN(10) PMTCTL(FILE)
   CMD: PMTCTL CTL(OBJ) COND((*EQ *CMD) (*EQ *)) NBRTRUE(*EQ 1)
   PGM: PMTCTL CTL(OBJ) COND((*EQ *PGM) (*EQ *)) NBRTRUE(*EQ 1)
   FILE: PMTCTL CTL(OBJ) COND((*EQ *FILE) (*EQ *)) NBRTRUE(*EQ 1)
```

El fuente para el programa de salida se muestra a continuación:

```
   PGM PARM(&CMD &PARMVAL &RTNVAL)
   DCL &CMD *CHAR 20                /* Nombre del mandato y parámetro  */
   DCL &PARMVAL *CHAR 20            /* Valor del parámetro             */
   DCL &RTNVAL *CHAR 32             /* Valor de retorno                */
   DCL &OBJNAM *CHAR 10             /* Nombre del objeto               */
   DCL &OBJLIB *CHAR 10             /* Tipo del objeto                 */
   CHGVAR &OBJNAM %SST(&PARMVAL 1 10)
   CHGVAR &OBJLIB %SST(&PARMVAL 11 10)
   CHGVAR &RTNVAL '*'               /* Valor devuelto si error         */
   CHKOBJ &OBJLIB/&OBJNAM *CMD      /* Consultar si existe mandato     */
   MONMSG CPF9801 EXEC(GOTO NOTCMD) /* Saltar si no hay mandato        */
   CHGVAR &RTNVAL '*CMD'            /* El objeto indicado es un mandato*/
   RETURN                           /* Salir                           */
   NOTCMD:
   CHKOBJ &OBJLIB/&OBJNAM *PGM      /* Consultar si existe programa    */
   MONMSG CPF9801 EXEC(GOTO NOTPGM) /* Saltar si no hay programa       */
   CHGVAR &RTNVAL '*PGM'            /* El obj. indicado es un programa */
   RETURN                           /* Salir                           */
   NOTPGM:
   CHKOBJ &OBJLIB/&OBJNAM *FILE     /* Consultar si existe archivo     */
   MONMSG CPF9801 EXEC(RETURN)      /* Salir si no hay archivo         */
   CHGVAR &RTNVAL '*FILE'           /* El objeto indicado es un archivo*/
```

```
ENDPGM
```

9.5.2 Parámetros Adicionales

Puede especificar que un parámetro que no se utiliza frecuentemente no se solicite a menos que el usuario pida parámetros adicionales pulsando una tecla de función durante la solicitud. Esto se lleva a cabo especificando PMTCTL(*PMTRQS) en la sentencia PARM para el parámetro. Al solicitar un mandato, los parámetros que tengan PMTCTL(*PMTRQS) codificado no se solicitarán a menos que se haya especificado un valor para ellos o que el usuario pulse F10 para pedir los parámetros adicionales.

La solicitud visualiza una línea de separación delante de los parámetros con PMTCTL(*PMTRQS) para distinguirlos de los demás parámetros. Por omisión, todos los parámetros con PMTCTL(*PMTRQS) se solicitan los últimos, aunque no estén definidos en este orden en el fuente de definición del mandato. Puede alterarse temporalmente esto especificando un número de solicitud relativo en la palabra clave PROMPT. Sin embargo, de este modo puede resultar difícil distinguir qué parámetros se añadieron a la solicitud cuando se pulsó F10.

9.6 Utilización de Parámetros Clave y un Programa de Alteración Temporal de Solicitudes

El programa de alteración temporal de solicitudes permite visualizar los valores actuales en lugar de los valores por omisión cuando se solicita un mandato.

Si se define un programa de alteración temporal de solicitud para un mandato, los resultados de la llamada al programa de alteración temporal de solicitudes pueden verse de las dos formas siguientes:

- Teclee el nombre del mandato sin parámetros en cualquier línea de mandatos y pulse F4= Solicitud. La pantalla siguiente muestra los parámetros clave para el mandato. Los parámetros clave son parámetros, tales como el nombre de un objeto, que identifican de forma exclusiva al objeto.

  Rellene todos los campos mostrados y pulse la tecla Intro. La siguiente pantalla muestra todos los parámetros del mandato, y los campos del parámetro que no son campos de parámetros clave contienen valores actuales en lugar de los valores por omisión (tales como *SAME y *PRV).

  Por ejemplo si teclea **CHGLIB** en una línea de mandatos y pulsa F4= Solicitud, verá solamente el parámetro Biblioteca. Si a continuación teclea ***CURLIB** y pulsa la tecla Intro, se visualizan los valores actuales de la biblioteca actual.

- Teclee el nombre del mandato y los valores de todos los parámetros clave en cualquier línea de mandatos. Pulse F4= Solicitud. La pantalla siguiente muestra todos los parámetros del mandato y los campos del parámetro que no son campos de parámetros clave contendrán valores actuales en lugar de los valores por omisión (tales como *SAME y *PRV).

  Por ejemplo si se teclea **CHGLIB LIB(*CURLIB)** en una línea de mandatos y se pulsa F4= Solicitud, se visualizan los valores actuales de la biblioteca actual.

Cuando se pulsa F10=Parámetros adicionales, se visualizan los parámetros definidos con PMTCTL(*PMTRQS) con sus valores actuales. Para obtener más información sobre parámetros adicionales, véase el apartado "Parámetros Adicionales" en el tema 9.5.2.

Para salir de la solicitud de mandatos, pulse F3=Salir.

Subtemas

9.6.1 Procedimiento para Utilizar Programas de Alteración Temporal de Solicitudes

Para utilizar un programa de alteración temporal de solicitudes, efectúe lo siguiente:

1. Especifique los parámetros que serán clave en la sentencia PARM en el fuente de definición de mandatos. Para obtener información sobre el parámetro KEYPARM, véase el apartado siguiente, "Identificación de Parámetros Clave" en el tema 9.6.1.1.

2. Escriba un programa de alteración temporal de solicitudes. Para obtener información sobre la creación de programas de alteración temporal de solicitudes, véase el apartado "Creación de un Programa de Alteración Temporal de Solicitudes" en el tema 9.6.1.2.

3. Especifique el nombre del programa de alteración temporal de solicitudes en el programa PMTOVRPGM cuando cree o modifique el mandato. Para obtener información sobre la creación o la modificación de mandatos que utilizan el programa de alteración de solicitudes, véase el apartado "Especificación del Programa de Alteración Temporal de Solicitudes al Crear o Modificar Mandatos" en el tema 9.6.1.3.

Subtemas
9.6.1.1 Identificación de Parámetros Clave
9.6.1.2 Creación de un Programa de Alteración Temporal de Solicitudes
9.6.1.3 Especificación del Programa de Alteración Temporal de Solicitudes al Crear o Modificar Mandatos

9.6.1.1 Identificación de Parámetros Clave

El número de parámetros clave debe limitarse al número de parámetros necesarios para definir de forma exclusiva el objeto que se va a modificar.

Para asegurarse de que un parámetro clave está codificado correctamente en el fuente de definición de mandatos, efectúe lo siguiente:

- Especifique KEYPARM(*YES) en la sentencia PARM en el fuente de definición de mandatos.
- Defina todos los parámetros que especifiquen KEYPARM(*YES) antes que todos los parámetros que especifican KEYPARM(*NO).

  Nota: Si una sentencia PARM especifica KEYPARM(*YES) detrás de una sentencia PARM que especifica a KEYPARM(*NO), el parámetro no se trata como parámetro clave y se emite un mensaje de aviso.
- No especifique un valor MAX mayor que uno en la sentencia PARM.
- No especifique un valor MAX superior a uno para los elementos ELEM asociados con los parámetros clave.
- No especifique *PMTRQS o una sentencia de control de solicitud para la palabra clave PMTCTL en la sentencia PARM.
- coloque los parámetros clave en el fuente de definición de mandatos en el mismo orden en que desea que aparezcan cuando se soliciten.

9.6.1.2 Creación de un Programa de Alteración Temporal de Solicitudes

Un programa de alteración temporal necesita que se le pase cierta información para devolver valores actuales cuando se solicita un mandato. Debe tener en cuenta tanto la información que se pasa como los valores devueltos cuando escriba un programa de alteración temporal de solicitudes.

Para ver un ejemplo de fuente CL para un programa de alteración temporal de solicitudes, consulte el apartado "Ejemplo de CL para la Utilización del Programa de Alteración Temporal de Solicitudes" en el tema 9.6.2.

Parámetros Pasados al Programa de Alteración Temporal de Solicitudes: Al programa de alteración temporal de solicitudes se le pasan los siguientes parámetros:

- Un campo de 20 caracteres. Los 10 primeros caracteres del campo contienen el nombre del mandato, mientras que los 10 últimos contienen el nombre de la biblioteca.

- Un valor para cada parámetro clave, si lo hay. Si se define más de un parámetro clave, los valores se pasan en el orden en que se han definido los parámetros en el fuente de definición de mandatos.

- Un espacio de 5700 bytes para colocar la serie de mandatos creada por el programa de alteración temporal de solicitudes. Los dos primeros bytes de este campo contiene la longitud de la serie de mandatos que se devuelve. A continuación figura la serie de mandatos real.

  Por ejemplo, si se definen dos parámetros para un mandato, se pasan cuatro parámetros al programa de alteración temporal de solicitudes tal como se muestra a continuación:

  - un parámetro para el mandato
  - dos parámetros para los parámetros clave
  - un parámetro para el espacio de la serie del mandato

Información Devuelta desde el Programa de Alteración Temporal de Solicitudes: En función de los valores pasados, el programa de alteración temporal de solicitudes recupera los valores actuales de los parámetros que no son clave. Estos valores se colocan en una serie de mandatos, donde se determina y se devuelve la longitud de la serie.

Siga las directrices siguientes para asegurarse de que su serie de mandatos está definida correctamente:

- Utilice el formato de palabra clave para la serie de mandatos igual que lo haría en la línea de mandatos.

- No incluya el nombre de mandato y los parámetros clave en la serie de mandato.

- Delante de cada palabra clave, coloque un carácter de solicitud selectiva para definir cómo se visualizará el parámetro y qué valor se pasará al CPP. Para obtener información sobre la utilización de caracteres de solicitud selectiva, véase el apartado "Solicitud Selectiva de Mandatos CL" en el tema 6.5.2.

  Cuando utilice solicitudes selectivas, haga lo siguiente:

  - Si define un parámetro como MIN(1) en el fuente de definición del mandato (es decir, se necesita el parámetro) debe utilizar el carácter de solicitud selectivo **??** para esta palabra clave en la serie del mandato desde el programa de alteración temporal de solicitud.
  - No utilice el carácter de solicitud selectivo **?-** en la serie de mandato del programa de alteración temporal de solicitudes.

  En el ejemplo siguiente se muestra una serie de mandatos devuelta de un programa de alteración temporal de solicitudes:

```
  ??Número(123456) ?<Calificador(CLIB/CFILE) ?<LIST(ITEM1 ITEM2 ITEM3) ?<TEXT('Archivo de Carol')
```

- Asegúrese de que el valor especificado en los dos primeros bytes del espacio pasado al programa es la longitud real de la serie de mandatos, y de que no incluye los dos primeros bytes de espacio.

```
¦ 0024 ¦ ??NAME(CAROL) ?□FILE(CHWLIB/CJWFILE) . . . ¦ . . . ¦
+-----------------------------------------------------------+
  ¦
  ¦      +--------  Serie Mandato   --------+

Longitud de la serie del mandato (en hexadecimal)
```

- Incluya solamente los parámetros de la serie de mandatos cuyos valores actuales desea visualizar cuando se solicita el mandato. En el caso de los parámetros no incluidos en la serie de mandatos, se visualizan sus valores por omisión.

- Utilice el formato de carácter para los número que aparezcan en la serie de mandatos. No utilice el formato binario o empaquetado. No incluya números hexadecimales en la serie de mandatos.

- No ponga espacios en blanco entre la biblioteca y el calificador o entre el calificador y el objeto. Por ejemplo:

  ??KWD1(biblioteca/objeto) No válido
  ??KWD1(biblioteca/objeto) No válido
  ??KWD1(biblioteca/objeto) Válido
  ??KWD1(biblioteca/objeto) Válido

- Si utiliza valores especiales o valores únicos, asegúrese que se conviertan a los valores origen definidos en el fuente de definición de mandatos.

  Por ejemplo, una palabra clave tiene un valor especial definido como SPCVAL(*SPECIAL *) en el fuente de definición de mandatos. *SPECIAL es el valor origen y * es el valor destino. Cuando se recupera el valor actual para esta palabra clave, * es el valor recuperado pero *SPECIAL debe aparecer en la serie de mandatos devuelta desde el programa de alteración temporal de solicitudes. El valor origen correcto debe colocarse en la serie de mandatos, puesto que más de un valor especial o único puede tener el mismo valor destino. Por ejemplo, si se especifica **KWD1 SPCVAL((*SPC *) (*SPECIAL *))**, el programa de alteración temporal de solicitudes debe determinar si * es el valor destino para *SPC o para *SPECIAL.

- Defina la longitud de los campos utilizados para recuperar el texto de la forma siguiente:

  ```
  (2*(longitud de campo definida en el fuente de definición de mandatos)) + 2
  ```

  Esta longitud permite utilizar el número máximo de comillas permitidas en el campo de texto. Por ejemplo, si se define el parámetro TEXT del mandato CHGxxx en el fuente de definición del mandato como LEN(50), el parámetro se declara como CHAR(102) en su programa de alteración temporal de solicitudes. Para ver un ejemplo de cómo definir la longitud de los campos utilizados para recuperar texto, consulte el apartado "Ejemplo de CL para la Utilización del Programa de Alteración Temporal de Solicitudes" en el tema 9.6.2.

  Si el parámetro para un campo de texto no está definido correctamente en el programa de alteración temporal de solicitudes y la serie de texto que recupera el programa de alteración temporal de solicitud contiene una comilla, el mandato no se solicita correctamente.

- Asegúrese que los apóstrofos intercalados están definidos como si fuesen comillas; por ejemplo:

  ```
  ?<TEXT('Carol"s library')
  ```

Algunos mandatos pueden ejecutarse solamente en ciertas modalidades (como DEBUG) o estados de trabajo (como *BATCH), pero pueden solicitarse desde otras modalidades o estados de trabajo. Cuando se solicita el mandato, se llama al programa de alteración temporal de solicitudes independientemente del entorno del usuario. Si se llama al programa de alteración temporal de solicitudes en una modalidad o entorno que no es válido para el mandato, se visualizan los valores por omisión para el mandato y se devuelve un valor 0 para la longitud. Como ejemplos de esta condición, pueden citarse los mandatos de depuración Cambiar Modalidad de Depuración (CHGDBG) y Añadir Programa (ADDPGM) cuando no se trabaja en modalidad de depuración.

Permitir Errores en un Programa de Alteración Temporal de Solicitudes: Si el programa de alteración temporal de solicitudes detecta un error, debe hacer lo siguiente:

- Establecer la longitud de la serie de mandatos a cero de modo que se visualicen los valores por omisión en lugar de los valores actuales cuando se solicita el mandato.

- Enviar un mensaje de diagnóstico a la llamada anterior.

- Enviar el mensaje de escape CPF0011.

Por ejemplo, si necesita un mensaje que diga que no existe una biblioteca, añada una descripción de mensaje parecida a la siguiente:

```
ADDMSGD      MSG('Biblioteca &2 no existe') +
             MSGID(USR0012) +
             MSGF(QGPL/ACTMSG) +
             SEV(40) +
             FMT((*CHAR 4) (*CHAR 10))
```

Nota: La variable de sustitución &1 no está en el mensaje pero está definida en el parámetro FMT como 4 caracteres. &1 está reservada para utilizar por el sistema, y debe ser siempre de 4 caracteres. Si la variable de sustitución &1 es la única variable de sustitución definida en el mensaje, compruebe que el cuarto byte de los datos del mensaje no contienen un blanco cuando envíe el mensaje. El cuarto byte lo utiliza el sistema para gestionar mensajes durante el proceso y la solicitud de mandatos.

Este mensaje puede enviarse al programa que ha llamado al programa de alteración temporal de solicitudes especificando en éste último lo siguiente:

```
SNDPGMMSG    MSGID(USR0012) MSGF(QGPL/ACTMSG) +
             MSGDTA('0000' || &libname) MSGTYPE(*DIAG)
```

Después de que el programa de alteración temporal de solicitudes envíe todos los mensajes de diagnóstico necesarios, debe enviarse el mensaje CPF0011. Para enviar este mensaje, utilice el mandato Enviar Mensaje de Programa (SNDPGMMSG) de la forma siguiente:

```
SNDPGMMSG    MSGID(CPF0011) MSGF(QCPFMSG) +
             MSGTYPE(*ESCAPE)
```

Cuando se recibe el mensaje CPF0011, se envía el mensaje CPD680A al programa de llamada y se visualiza en la pantalla de solicitud para indicar que se han hallado errores. Todos los mensajes de diagnóstico se colocan en las anotaciones de trabajo del usuario.

9.6.1.3 Especificación del Programa de Alteración Temporal de Solicitudes al Crear o Modificar Mandatos

Si va a utilizar un programa de alteración temporal de solicitudes para un mandato que desee crear, especifique el nombre del programa cuando utilice el mandato Crear Mandato (CRTCMD). También puede especificar el nombre del programa cuando modifique el mandato utilizando el mandato Cambiar Mandato (CHGCMD). En ambos mandatos, especifique el nombre del programa de alteración temporal de solicitudes en el parámetro PMTOVRPGM.

Si se definen los parámetros clave en el fuente de definición de mandatos pero el programa de alteración temporal de solicitudes no está especificado cuando se crea o se cambia el mandato, se produce como resultado el mensaje de aviso CPD029B. Los parámetros clave se pasan por alto y, cuando se solicita el mandato, se visualiza utilizando los valores por omisión especificados en el fuente de definición de mandatos.

A veces se especifica un programa de alteración temporal de solicitudes cuando se crea un mandato pero no se han definido parámetros clave en el fuente de definición de mandatos. En estos casos, se llama al programa de alteración temporal de solicitudes antes de que se solicite el mandato; se envía el mensaje informativo CPD029A cuando se crea o cambia el mandato.

9.6.2 Ejemplo de CL para la Utilización del Programa de Alteración Temporal de Solicitudes

En el ejemplo siguiente se muestra el fuente de un mandato y el programa de alteración temporal de solicitudes. Este mandato permite cambiar la propiedad y el texto de descripción de una biblioteca. El programa de alteración temporal de solicitudes para este mandato recibe el nombre de la biblioteca, recupera el valor actual del propietario de la biblioteca y el texto de la descripción y, por último, coloca estos valores en una serie de mandatos y la devuelve.

Subtemas

9.6.2.1 Ejemplo de Fuente del Mandato

```
  CHGLIBATR: CMD   PROMPT('Cambiar Atributos de Biblioteca')
             PARM  KWD(LIB) +
                     TYPE(*CHAR) MIN(1) MAX(1) LEN(10) +
                     KEYPARM(*YES) +
                     PROMPT('Biblioteca a cambiar')
             PARM  KWD(OWNER) +
                     TYPE(*CHAR) LEN(10) MIN(0) MAX(1) +
                     KEYPARM(*NO) +
                     PROMPT('Propietario de biblioteca')
             PARM  KWD(TEXT) +
                     TYPE(*CHAR) MIN(0) MAX(1) LEN(50) +
                     KEYPARM(*NO) +
                     PROMPT('Texto de descripción')
```

9.6.2.2 Ejemplo de Programa de Alteración Temporal de Solicitudes

```
 PGM PARM(&cmdname &keyparm1 &rtnstring)
/*****************************************************************/
/*                                                               */
/*  Declaraciones de parámetros pasados al programa de alteración    */
/*  temporal de solicitudes                                      */
/*                                                               */
/*****************************************************************/
 DCL  VAR(&cmdname)   TYPE(*CHAR) LEN(20)
 DCL  VAR(&keyparm1)   TYPE(*CHAR) LEN(10)
 DCL  VAR(&rtnstring) TYPE(*CHAR) LEN(5700)

/*****************************************************************/
/*                                                               */
/* Devolver declaración de la estructura de la serie de mandatos    */
/*                                                               */
/*****************************************************************/

                      /* Longitud de la serie de mandatos generada  */
 DCL  VAR(&stringlen) TYPE(*DEC)  LEN(5 0) VALUE(131)
 DCL  VAR(&binlen)    TYPE(*CHAR) LEN(2)
                      /* palabra clave OWNER                        */
 DCL  VAR(&ownerkwd)  TYPE(*CHAR) LEN(8)  VALUE('?<OWNER(')
 DCL  VAR(&name)      TYPE(*CHAR) LEN(10)
                      /* palabra clave TEXT                         */
 DCL  VAR(&textkwd)   TYPE(*CHAR) LEN(8)  VALUE(' ?<TEXT(')
 DCL  VAR(&descript)  TYPE(*CHAR) LEN(102)

/*****************************************************************/
/*                                                               */
/* Variables relacionadas con declaraciones de series de mandatos   */
/*                                                               */
/*****************************************************************/
 DCL  VAR(&quote)      TYPE(*CHAR) LEN(1) VALUE('''')
 DCL  VAR(&closparen)  TYPE(*CHAR) LEN(1) VALUE(')')

/*****************************************************************/
/*                                                               */
/*                  Inicio del código ejecutable                 */
/*                                                               */
/*****************************************************************/
/*****************************************************************/
/*                                                               */
/* Supervisar excepciones                                        */
/*                                                               */
/*****************************************************************/
    MONMSG  MSGID(CPF0000) +
            EXEC(GOTO CMDLBL(error))

/*****************************************************************/
/*                                                               */
/* Recuperar propietario y texto descrito para la bibl especificada */
/* en el parámetro LIB. Nota: Este programa presupone que no hay    */
/* apóstrofos en TEXTO, como (Carol''s)                          */
/*                                                               */
/*****************************************************************/
    RTVOBJD OBJ(&keyparm1) OBJTYPE(*LIB) OWNER(&name) TEXT(&descript)

    CHGVAR VAR(%BIN(&binlen)) VALUE(&stringlen)

/*****************************************************************/
/*                                                               */
/* Construir la serie del mandato                                */
/*                                                               */
/*****************************************************************/
    CHGVAR VAR(&rtnstring) VALUE(&binlen)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &ownerkwd)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &name)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &closparen)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &textkwd)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &quote)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &descript)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &quote)
    CHGVAR VAR(&rtnstring) VALUE(&rtnstring *TCAT &closparen)

    GOTO CMDLBL(pgmend)

    CHGVAR VAR(&stringlen) VALUE(0)
    CHGVAR VAR(%BIN(&binlen)) VALUE(&stringlen)
    CHGVAR VAR(&rtnstring) VALUE(&binlen)

/*****************************************************************/
```

```
/*                                                                  */
/* Enviar mensaje(s) de error                                       */
/*                                                                  */
/* NOTA: Si desea enviar un mensaje diagnóstico y CPF0011, deberá   */
/*       entrar un ID válido de mensaje de error en el parámetro    */
/*       MSGID y un archivo de mensajes válido en el parámetro MSGF */
/*       para el primer mandato SNGPGMMSG listado anteriormente.    */
/*       Si no desea enviar un mensaje de diagnóstico, no incluya   */
/*       el primer SNDPGMMSG. Sin embargo, en condiciones de error, */
/*       SIEMPRE debe enviar CPF0011 de modo que debe incluirse el  */
/*       segundo mandato SNDPGMMSG en su programa.                  */
/*                                                                  */
/********************************************************************/
    SNDPGMMSG MSGID(XXXXXXX) MSGF(MSGLIB/MSGFILE) MSGTYPE(*DIAG)
    SNDPGMMSG MSGID(CPF0011) MSGF(QCPFMSG) MSGTYPE(*ESCAPE)

PGMEND:
ENDPGM
```

9.7 Creación de Mandatos

Después de definir el mandato mediante las sentencias de definición de mandatos, debe utilizar el mandato Crear Mandato (CRTCMD) para crearlo. Además de especificar el nombre del mandato, el nombre de la biblioteca y el nombre del programa de proceso de mandatos para CL o para lenguajes de alto nivel (HLL), así como el miembro fuente, el archivo fuente, el entorno del mandato y el programa de salida para REXX, puede definir los siguientes atributos del mandato:

- La comprobación de validez utilizada por el mandato
- Las modalidades en la que se puede ejecutar el mandato
  - Producción
  - Depuración
  - Servicio
- Dónde puede utilizarse el mandato
  - Trabajo por lotes
  - Trabajo interactivo
  - Módulo CL ILE es un trabajo por lotes
  - Programa CL en un trabajo por lotes
  - Módulo CL ILE es un trabajo interactivo
  - Programa CL en un trabajo interactivo
  - Procedimiento REXX en un trabajo por lotes
  - Procedimiento REXX en un trabajo interactivo
  - Como un mandato procesado interpretativamente por el sistema a través de una llamada a QCMDEXC (véase el Capítulo 6 para obtener información sobre QCMDEXC)
- El número máximo de parámetros que pueden especificarse por posición
- El archivo de mensajes que contiene el texto de solicitud
- El grupo de paneles de ayuda que se utilizan como ayuda para parámetros que se pueden solicitar
- El nombre del identificador de ayuda para el módulo de ayuda general usado en este mandato
- El archivo de mensajes que contiene los mensajes identificados en la sentencia DEP
- La biblioteca actual que estará activa durante el proceso del mandato
- La biblioteca del producto que estará activa durante el proceso del mandato
- Si un mandato existente con el mismo nombre, tipo y biblioteca se sustituye si se especifica REPLACE(*YES).
- La autorización otorgada al público para el mandato y su descripción
- El texto que describe brevemente el mandato y su función

Para mandatos con CPP REXX, también puede especificar lo siguiente:

- El entorno inicial del mandato para manejar los mandatos cuando arranca el procedimiento
- Programas de salida para controlar la ejecución del procedimiento

El ejemplo siguiente define un mandato denominado ORDENTRY para llamar a una aplicación de entrada de pedidos. El mandato CRTCMD define los atributos precedentes para ORDENTRY y crea el mandato que utilizando las definiciones de parámetro contenidas en el miembro ORDENTRY del archivo fuente QCMDSRC suministrado por IBM. ORDENTRY contiene la sentencia PARM utilizada en el ejemplo del apartado "Ejemplo de Definición de un Parámetro" en el tema 9.2.2.5.

```
  CRTCMD     CMD(DSTPRODLB/ORDENTRY) +
             PGM(*LIBL/ORDENT) +
             TEXT('Llamadas a aplicación de entrada de pedidos')
```

El mandato resultante es:

```
  ORDENTRY  OETYPE(valor)
```

donde el valor puede ser DAILY, WEEKLY o MONTHLY.

Una vez que se ha creado un mandato, puede:

- Visualizar los atributos del mandato utilizando el mandato Visualizar Mandato (DSPCMD)

- Cambiar los atributos del mandato utilizando el mandato Cambiar Mandato (CHGCMD)

- Suprimir el mandato utilizando el mandato Suprimir Mandato (DLTCMD)

Subtemas

9.7.1 Listado Fuente de Definición de Mandatos

Cuando se crea un mandato, se genera una lista fuente. A continuación se muestra un ejemplo de lista fuente. Los números que aparecen hacen referencia a las descripciones que se encuentran después de la lista.

```
   5763SS1 V3R1M0  940909 1       Definición Mandatos         DSTPRODLB/ORDENTRY    26/05/94 10:35:32 2 Pág

   Nombre mandato. . . . . . . . . . . . . . . . :   ORDENTRY
     Biblioteca  . . . . . . . . . . . . . . . . :     DSTPRODLB
   Programa de proceso de mandatos . . . . . . . :   ORDENT      4
     Biblioteca  . . . . . . . . . . . . . . . . :     DSTPRODLB
   Archivo fuente  . . . . . . . . . . . . . . . :   QCMDSRC
     Biblioteca  . . . . . . . . . . . . . . . . :     QGPL
   Miembro de archivo fuente . . . . . . . . . . :   ORDENTRY     26/05/94  10:33:32
   Programa comprobador de validez . . . . . . . :   *NONE
   Modalidad en que es válido  . . . . . . . . . :   *PROD
                                                     *DEBUG
                                                     *SERVICE
   Entorno permitido . . . . . . . . . . . . . . :   *IREXX
                                                     *BREXX
                                                     *BPGM
                                                     *IPGM
                                                     *EXEC
                                                     *INTERACT
                                                     *BATCH
                                                     *BMOD
                                                     *IMOD
   Permitir usuario limitado . . . . . . . . . . :   *NO
   Máx parámetros posicionales . . . . . . . . . :   *NOMAX
   Archivo de solicitudes  . . . . . . . . . . . :   *NONE
   Archivo de mensajes . . . . . . . . . . . . . :   QCPFMSG
     Biblioteca. . . . . . . . . . . . . . . . . :     *LIBL
   Autorización  . . . . . . . . . . . . . . . . :   *USE
   Sustituir mandato . . . . . . . . . . . . . . :   *YES
   Texto . . . . . . . . . . . . . . . . . . . . :   Llama a pedido
   Nombre libro ayuda  . . . . . . . . . . . . . :   *NONE
   Estantería ayuda  . . . . . . . . . . . . . . :   *NONE
   Grupo paneles ayuda . . . . . . . . . . . . . :   HLPORDEN
     Biblioteca. . . . . . . . . . . . . . . . . :     *LIBL
   Identificador ayuda . . . . . . . . . . . . . :   HIDORDEN
   Índice de búsqueda de ayuda . . . . . . . . . :   *NONE
   Biblioteca actual . . . . . . . . . . . . . . :   *NOCHG
   Biblioteca de productos . . . . . . . . . . . :   *NOCHG
   Programa alteración temporal solicitudes  . . :   *NONE
   Compilador  . . . . . . . . . . . . . . . . . :   Compilador Definición Mandatos AS/400 IBM  5

                                    Fuente Definición Mandatos
    6
   SEQNBR  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+.
      100-               CMD        PROMPT('Mandato entrada pedidos')
   7
      200-               PARM       KWD(OETYPE) TYPE(*CHAR) RSTD(*YES) +
      300                             VALUES(DAILY WEEKLY MONTHLY) MIN(1) +
      400                             PROMPT('Tipo entrada pedidos:')
                                          * * * * *   F I N   D E   F U E N T E   * * * * *

   5763SS1 V3R1M0  940909                Definición Mandatos              DSTPRODLB/ORDENTRY     26/05/94
                                         Referencias Cruzadas

   Palabras clave definidas  9
   Pal Clave            Número          Definida     Referencias
   OETYPE                 001               200
                 * * * * *   F I N   D E   R E F E R E N C I A S   C R U Z A D A S   * * * *

   5763SS1 V3R1M0  940909                Definición Mandatos              DSTPRODLB/ORDENTRY     26/05/94
                                          Mensajes Finales

   ID                     Número
   Mensaje                Secuencia       Grav       Texto  10

                                        Resumen Mensajes
   Total         Info        Error
       0            0           0    11
    * CPC0202                             00        Mandato ORDENTRY creado en bibliot. DSTPRODLB.  12
                                  * * * * *   F I N   D E   C O M P I L A C I O N   * * * * *
```

Título:

1 Número de programa, versión, release, nivel de modificación y fecha de

OS/400.

2 Fecha y la hora de esta ejecución.

3 Número de página del listado.

Prólogo:

4 Valores del parámetro especificados (o valores por omisión si no se especificaron) en el mandato CRTCMD. Si el fuente no está en un archivo de base de datos, se omiten el nombre del miembro, la fecha y la hora.

5 Nombre del compilador de definición del mandato de crear.

Fuente:

6 Número de secuencia de líneas (registros) en el archivo fuente. Un guión a continuación de un número de secuencia indica que una sentencia fuente empieza en dicho número de secuencia. La ausencia de guión indica que una sentencia es la continuación de la sentencia anterior. Por ejemplo, una sentencia fuente PARM empieza en el número de secuencia 200 y continúa en los números de secuencia 300 y 400. (Observe el carácter de continuación + en la sentencia PARM en el número de secuencia 200 y 300.)

Las sentencias fuente de comentario se manejan como cualquier otra sentencia fuente y tienen números de secuencia.

Consulte el manual *DB2/400 Programación de la base de datos* para obtener información sobre cómo se asignan los números de secuencia.

7 Sentencias fuente.

8 Última fecha en que se cambió o añadió la sentencia fuente. Si la sentencia no se ha cambiado o añadido, no se muestra fecha. Si el fuente no está en un archivo de base de datos, se omite la fecha.

Si se encuentra un error durante el proceso de las sentencias de definición del mandato y pueden rastrearse hasta una sentencia fuente específica, el mensaje de error se imprime inmediatamente después de la sentencia fuente. Un asterisco (*) indica que la línea contiene un mensaje de error. La línea contiene el identificador del mensaje, la gravedad y el texto del mensaje.

Para más información sobre los errores de definición de mandatos, véase el apartado "Errores Encontrados al Procesar Sentencias de Definición de Mandatos" en el tema 9.7.2.

Referencias cruzadas:

9 La tabla de palabras clave es una lista de referencias cruzadas de las palabras clave definidas de forma válida en la definición del mandato. En la tabla figuran la palabra clave, la posición de la palabra clave en el mandato, el número de secuencia de la sentencia donde está definida la palabra clave y los números de secuencia de las sentencias que hacen referencia a la palabra clave.

Si se definen etiquetas válidas en la definición del mandato, se facilita un listado de referencias cruzadas de las etiquetas (tabla de etiquetas). En la tabla figuran la etiqueta, el número de secuencia de la sentencia donde se define la etiqueta y los números de secuencia de las sentencias que hacen referencia a la etiqueta.

Mensajes:

10 Listado de los mensajes de error generales no listados en la sección fuente que se encontraron durante el proceso de las sentencias de definición del mandato, si las hay. Para cada mensaje, esta sección contiene el identificador de mensaje, el número de secuencia de donde se produjo el error, la gravedad y el mensaje.

Resumen de mensajes:

11 Resumen del número de mensajes emitidos durante el proceso de las sentencias de definición del mandato. Se da el número total junto con los totales por gravedad.

12 Se imprime un mensaje de terminación a continuación del resumen de

```
mensajes.
```

9.7.2 Errores Encontrados al Procesar Sentencias de Definición de Mandatos

Los tipos de errores que se detectan durante el proceso de las sentencias de definición de mandatos incluyen errores de sintaxis, referencias a palabras clave y etiquetas no definidas y sentencias que faltan. Si el compilador de definición de mandatos detecta los siguientes tipos de errores, el proceso de creación del mandato se detiene (los códigos de gravedad se pasan por alto).

- Errores de valores
- Errores de sintaxis

Aunque se encuentre un error que impida la creación del mandato, el compilador de definición de mandatos sigue comprobando si hay otros errores en el fuente. Los errores de sintaxis y de valores fijos impiden la comprobación final que identifica los errores en los nombres de usuario y los valores o referencias a palabras claves o etiquetas. La búsqueda de errores de sintaxis y de valores fijos continúa. Esto le permite ver y corregir tantos errores como sea posible antes de intentar crear de nuevo el mandato. Para corregir errores en las sentencias fuente, consulte el manual *DB2/400 Programación de la base de datos* o *Screen Design Aid User's Guide and Reference*, SC09-1340.

En la lista fuente de definición de mandatos, las condiciones de error que están relacionadas directamente con una sentencia fuente determinada aparecen después de este mandato. Consulte el apartado "Listado Fuente de Definición de Mandatos" en el tema 9.7.1 para ver un ejemplo de estos mensajes incorporados. Los mensajes que no están relacionados con una sentencia fuente específica pero que son más generales se listan en una sección de mensajes de la lista, no incorporados con sentencias fuente.

9.8 Visualización de una Definición de Mandato

Puede utilizar el mandato Visualizar Mandato (DSPCMD) para visualizar o imprimir los valores que se han especificado como parámetros en el mandato CRTCMD. Dicho mandato visualiza la siguiente información para sus mandatos o para los proporcionados por IBM:

- Nombre calificado del mandato. El nombre de la biblioteca es el nombre de la biblioteca en la que se encuentra el mandato visualizado.

- Nombre calificado del programa de proceso de mandatos. El nombre de la biblioteca es el nombre de la biblioteca en la que residía el programa de proceso de mandatos cuando se creó el mandato si el nombre de una biblioteca estaba especificado en el mandato CRTCMD o CHGCMD. Si no se especificó un nombre de biblioteca, se visualiza *LIBL como calificador de biblioteca. Si el CPP es un procedimiento REXX, se muestra *REXX.

- Nombre de archivo fuente calificado, si el archivo fuente era un archivo de base de datos. El nombre de biblioteca es el nombre de la biblioteca en la que se encontraba el archivo fuente cuando se procesó el mandato CRTCMD. Este campo está en blanco si el archivo fuente no era un archivo de base de datos.

- Nombre del miembro del archivo fuente, si el archivo fuente era un archivo fuente de base de datos.

- Si el CPP es un procedimiento REXX, se muestra la siguiente información:

  - Nombre de miembro del procedimiento REXX
  - Nombre calificado del archivo fuente REXX donde se encuentra el procedimiento REXX
  - Entorno del mandato REXX
  - Programas de salida REXX

- Nombre calificado del programa de comprobación de validez. El nombre de biblioteca es el nombre de la biblioteca en la que se encontraba el programa en el momento en que se creó el mandato, si el nombre de una biblioteca estaba especificado en el mandato CRTCMD o CHGCMD. Si no se especificó un nombre de biblioteca, se visualiza *LIBL como calificador de biblioteca.

- Modalidades válidas de operación.

- Entornos válidos en los que puede ejecutarse el mandato.

- Límite de posición para el mandato. Se visualiza *NOMAX si no existe límite posicional para el mandato.

- Nombre calificado del archivo de mensajes de solicitud. El nombre de biblioteca es el nombre de la biblioteca en la que se encontraba el archivo de mensajes cuando se ejecutó el mandato CRTCMD. Se visualiza *NONE si no existe ningún archivo de mensajes de solicitud para el mandato.

- Nombre calificado del archivo de mensajes para la sentencia DEP. Si se especificó un nombre de biblioteca para el archivo de mensajes cuando se creó el mandato, se visualiza dicho nombre de biblioteca. Si se utilizó la lista de bibliotecas cuando se creó el mandato, se visualiza *LIBL. Se visualiza *NONE si no existe ningún archivo de mensajes DEP para el mandato.

- Nombre calificado del grupo de paneles de ayuda.

- El nombre del identificador de ayuda para el mandato.

- Nombre calificado del programa de alteración temporal de solicitudes.

- Texto asociado al mandato. Se visualizan espacios en blanco si no existe texto para el mandato.

9.9 Consecuencias de Cambiar la Definición del Mandato de un Mandato en un Procedimiento o Programa

Cuando se crea un módulo o programa CL, las definiciones de mandato de los mandatos del procedimiento o programa se utilizan para generar el módulo o programa. Cuando se ejecuta el procedimiento o programa CL, también se utilizan las definiciones de mandato. Si especifica un nombre de biblioteca para el mandato en el procedimiento o programa CL, el mandato debe estar en la misma biblioteca cuando se crea el procedimiento y cuando se ejecuta. Si especifica *LIBL para el mandato en el procedimiento o programa CL, se encuentra el mandato, ambos en tiempo de creación y de ejecución, utilizando la lista de bibliotecas (*LIBL).

Puede efectuar las modificaciones que se indican a continuación en las sentencias de definición de un mandato sin necesidad de volver a crear los módulos y programas que utilizan el mandato. Algunas de estas modificaciones tienen lugar en el fuente de las sentencias de definición de mandatos, lo que requiere que el mandato se cree de nuevo. Otras modificaciones se pueden realizar con el mandato Cambiar Mandato (CHGCMD).

- Añadir un parámetro opcional en cualquier posición. La adición de un parámetro opcional antes del límite de posición puede afectar a cualquier procedimiento, programa o corriente de entrada por lotes que tenga los parámetros especificados en forma posicional.
- Cambiar las comprobaciones REL y RANGE para que sean menos restrictivas.
- Añadir nuevos valores especiales. Sin embargo, esto podría cambiar la acción del procedimiento o programa si el valor se especificara antes del cambio.
- Cambiar el orden de los parámetros. Sin embargo, el cambio del orden de los parámetros que precedan al límite posicional *afectará* a cualquier procedimiento, programa o corriente de entrada por lotes que tenga los parámetros especificados en forma posicional.
- Incrementar el número de elementos opcionales en una lista sencilla.
- Cambiar los valores por omisión. Sin embargo, esto puede afectar la operación del procedimiento o programa.
- Disminuir el número de elementos necesarios en una lista sencilla.
- Hacer que un parámetro necesario se convierta en opcional.
- Cambiar RSTD de *YES a *NO.
- Incrementar la longitud cuando se especifique FULL(*NO).
- Cambiar FULL de *YES a *NO.
- Cambiar el texto de PROMPT.
- Cambiar el valor ALLOW para que sea menos restrictivo.
- Cambiar el nombre del programa de proceso de mandatos si el nuevo programa de proceso de mandatos acepta el número y tipo correcto de parámetros.
- Cambiar el nombre de la comprobación de validez si la nueva comprobación de validez acepta el número y tipo correcto de parámetros.
- Cambiar la modalidad en la que el mandato puede ejecutarse siempre y cuando la nueva modalidad no afecte a la modalidad antigua del mismo mandato que se utiliza en un procedimiento o programa CL.
- Cambiar TYPE por un valor compatible y menos restrictivo. Por ejemplo, cambiar TYPE de *NAME a *CHAR.
- Cambiar el valor MAX por uno mayor que 1.
- Cambiar los valores PASSATR y VARY.

Puede efectuar los cambios siguientes en las sentencias de definición de un mandato según lo que se especificó en el procedimiento o programa CL en el que se utiliza dicho mandato:

- Eliminar un parámetro.
- Cambiar los valores RANGE y REL para que sean más restrictivos.
- Eliminar los valores especiales.
- Disminuir el número de elementos permitidos en una lista.
- Cambiar el valor TYPE para que sea más restrictivo o incompatible con

el valor TYPE original. Por ejemplo, cambie el valor TYPE de *CHAR a *NAME o cambie *PNAME a *CHAR.

- Añadir un parámetro SNGVAL que previamente era un elemento de lista.
- Cambiar el nombre de un parámetro opcional.
- Eliminar un valor de una lista de valores.
- Aumentar el número de elementos de la lista necesarios.
- Cambiar un parámetro SNGVAL por uno SPCVAL.
- Cambiar una lista sencilla por una mixta de elementos similares.
- Cambiar un parámetro opcional por una constante.
- Cambiar RTNVAL de *YES a *NO, o de *NO a *YES.
- Cambiar el valor de mayúsculas/minúsculas de *MIXED a *MONO.

Puede efectuar los cambios siguientes en las sentencias de definición del mandato, pero puede dar lugar a que el funcionamiento del procedimiento o programa que utiliza el mandato sea diferente:

- Cambiar el significado de un valor.
- Cambiar el valor por omisión.
- Cambiar un parámetro SNGVAL por uno SPCVAL.
- Cambiar un valor por un parámetro SNGVAL.
- Cambiar una lista por una lista dentro de una lista.
- Cambiar el valor de mayúsculas/minúsculas de *MIXED a *MONO.

Los cambios siguientes en las sentencias de definición del mandato obligan a que el procedimiento o programa que utiliza el mandato tenga que crearse de nuevo.

- Adición de un nuevo parámetro necesario.
- Eliminación de un parámetro necesario.
- Cambio del nombre de un parámetro necesario.
- Cambio de un parámetro necesario por una constante.
- Cambio del programa de proceso de mandatos por o desde *REXX.

Además, si se especifica *LIBL como el calificador en el nombre del programa de proceso de mandatos o de la comprobación de validez cuando se crea o cambia el mandato, se puede trasladar el programa de proceso del mandato o la comprobación de validez a otra biblioteca de la lista de bibliotecas sin cambiar las sentencias de definición del mandato.

Subtemas

9.9.1 Cambio de los Valores por Omisión

Puede cambiar el valor por omisión de una palabra clave de mandato utilizando el mandato Cambiar Valor por Omisión de Mandato (CHGCMDDFT) descrito en el manual CL Reference. La palabra clave debe tener un valor por omisión para poder especificar uno nuevo. El mandato que se modifica puede ser uno suministrado por IBM o uno escrito por el usuario. Deben tomarse precauciones a la hora de cambiar los valores por omisión de mandatos proporcionados por IBM. A continuación se facilitan una serie de recomendaciones que hay que tener en cuenta al cambiar los valores por omisión:

1. Si el mandato que va a cambiarse es un mandato suministrado por IBM, debe utilizarse el mandato Crear Objeto Duplicado (CRTDUPOBJ) para crear un duplicado del mandato que va a cambiarse en una biblioteca de usuario. Esto permite, si es necesario, que otros usuarios del sistema utilicen los valores por omisión suministrados por IBM.

   Utilice el mandato Cambiar Lista de Bibliotecas del Sistema (CHGSYSLIBL) para trasladar la biblioteca de usuario delante de QSYS o de cualquier otra biblioteca suministrada por el sistema en la lista de bibliotecas. Esto permitirá que otros usuarios utilicen el mandato cambiado sin utilizar el calificador de biblioteca.

   Los cambios de mandatos que se necesitan de forma generalizada en el sistema deben efectuarse en una biblioteca de usuario y el nombre de ésta debe añadirse al valor del sistema QSYSLIBL delante de QSYS. El mandato modificado se utiliza en todo el sistema. Si necesita ejecutar una aplicación utilizando el valor por omisión suministrado por IBM, puede hacerlo utilizando el mandato Cambiar Lista de Bibliotecas del Sistema (CHGSYSLIBL) para suprimir la biblioteca especial o el calificador de biblioteca de los mandatos afectados.

2. Cuando se instala un nuevo release de un programa bajo licencia, se sustituyen en la máquina todos los mandatos suministrados por IBM para el programa bajo licencia. Por lo tanto, deberá utilizar un programa CL para efectuar cambios a mandatos de modo que, cuando se instale un nuevo release, pueda ejecutar el programa CL para duplicar los nuevos mandatos con el fin de tomar cualquier palabra clave nueva y efectuar los cambios de los valores por omisión.

   Si un mandato suministrado por IBM tiene palabras clave nuevas, puede que la copia del mandato de un release anterior no se ejecute correctamente.

   A continuación se facilita un ejemplo de un programa CL utilizado para suprimir la versión antigua y crear el nuevo mandato cambiado:

```
PGM
DLTCMD USRQSYS/SIGNOFF
CRTDUPOBJ OBJ(SIGNOFF) FROMLIB(QSYS) OBJTYPE(*CMD) +
          TOLIB(USRQSYS)  NEWOBJ(*SAME)
CHGCMDDFT CMD(USRQSYS/SIGNOFF) NEWDFT('LOG(*LIST)')
 .
 .
Repita los mandatos DLTCMD, CRTDUPOBJ y CHGCMDDFT para cada mandato que
desee modificar
 .
 .
ENDPGM
```

Los pasos siguientes se pueden utilizar para crear la serie de mandatos NEWDFT para el mandato CHGCMDDFT. En este ejemplo se utiliza el mandato USRQSYS/CRTCLPGM.

1. Crear una copia duplicada del mandato que se va a modificar en una biblioteca de usuario con el mandato siguiente:

```
  CRTDUPOBJ OBJ(CRTCLPGM) FROMLIB(QSYS) OBJTYPE(*CMD) +
            TOLIB(USRQSYS)  NEWOBJ(*SAME)
```

2. Entrar el nombre del mandato que va a modificarse en un archivo fuente al que el Programa de Utilidad para Entrada del Fuente (SEU) hace referencia.

3. Pulsar F4 para llamar a la solicitud del mandato.

4. Entrar los valores por omisión nuevos para las palabras clave que van a cambiarse. En este ejemplo, se entran **AUT(*EXCLUDE)** y **TEXT('Isn't this nice text')**.

5. Las palabras clave necesarias no pueden tener un valor por omisión; sin embargo, con el fin de obtener la serie del mandato en el archivo

fuente, debe especificarse un valor válido para cada palabra clave necesaria. Especifique PGM1 para el parámetro PGM.

6. Pulsar la tecla Intro para situar la serie del mandato en el archivo fuente. La serie del mandato devuelta sería la siguiente:

```
  USRQSYS/CRTCLPGM PGM(PGM1) AUT(*EXCLUDE) +
  TEXT('Isn''t this nice text')
```

7. Eliminar las palabras clave necesarias de la serie del mandato:

```
  USRQSYS/CRTCLPGM  AUT(*EXCLUDE) +
  TEXT('Isn''t this nice text')
```

   Recuerde que sólo puede cambiar los parámetros, elementos o calificadores que tengan valores por omisión existentes. Si se especifica un valor para un parámetro, elemento o calificador que no tenga un valor por omisión existente, no se efectúan los cambios en los valores por omisión.

8. Insertar al principio CHGCMDDFT tal como se muestra:

```
  CHGCMDDFT USRQSYS/CRTCLPGM AUT(*EXCLUDE) +
  TEXT('Isn''t this nice text')
```

9. La entrada para la palabra clave NEWDFT debe entrecomillarse, tal como se muestra:

```
  CHGCMDDFT USRQSYS/CRTCLPGM 'AUT(*EXCLUDE) +
  TEXT('Isn''t this nice text')'
```

10. Puesto que hay apóstrofos intercalados en el valor NEWDFT, deben duplicarse para ejecutarse adecuadamente:

```
  CHGCMDDFT USRQSYS/CRTCLPGM 'AUT(*EXCLUDE) +
  TEXT(''Isn''''t this nice text'')'
```

11. Ahora, si pulsa F4 para llamar a la solicitud del mandato y luego F11 para pedir la solicitud de palabras clave, se verá la pantalla siguiente:

```
  Mandato . . . . . :   CMD         R   CRTCLPGM
    Biblioteca. . . . . :                 USRQSYS
  Nueva serie parám.  omisión :   NEWDFT      R   'AUT(*EXCLUDE)
  TEXT(''Isn''''t this nice text'')'
```

12. Ahora si pulsa la tecla Intro, la serie del mandato CHGCMDDFT es:

```
  CHGCMDDFT CMD(USRQSYS/CRTCLPGM) NEWDFT('AUT(*EXCLUDE) +
            TEXT(''Isn''''t  this nice text'')')
```

13. Pulse F1 para salir de SEU y crear y ejecutar el programa o procedimiento CL.

14. USRQSYS/CRTCLPGM tendrá los valores por omisión de **AUT *EXCLUDE** y **'Isn''t this nice text'** para TEXT.

Subtemas

9.9.1.1 Ejemplo 1

Para proporcionar un valor por omisión de *NOMAX para la palabra clave MAXMBRS del mandato CRTPF, haga lo siguiente:

```
     CRTPF FILE(FILE1) RCDLEN(96) MAXMBRS(1)
       .
       .
     CHGCMDDFT CMD(CRTPF) NEWDFT('MAXMBRS(*NOMAX)')
```

9.9.1.2 Ejemplo 2

Para proporcionar un valor por omisión de 10 para la palabra clave MAXMBRS del mandato CRTPF, haga lo siguiente:

```
CRTPF FILE(FILE1) RCDLEN(96) MAXMBRS(*NOMAX)
  .
  .
CHGCMDDFT CMD(CRTPF) NEWDFT('MAXMBRS(10)')
```

9.9.1.3 Ejemplo 3

Este ejemplo le permite proporcionar un valor por omisión de LIB001 para el primer calificador de la palabra clave SRCFILE y de FILE001 para el segundo calificador de la palabra clave SRCFILE del mandato CRTCLPGM. La palabra clave AUT tiene ahora como valor por omisión *EXCLUDE.

```
     CRTCLPGM PGM(PROGRAM1) SRCFILE(*LIBL/QCMDSRC)
       .
       .
     CHGCMDDFT CMD(CRTCLPGM) +
               NEWDFT('SRCFILE(LIB001/FILE001) AUT(*EXCLUDE)')
```

9.9.1.4 Ejemplo 4

Este ejemplo proporciona un valor por omisión de 'Isn''t this print text' para la palabra clave PRTTXT del mandato CHGJOB. Puesto que la palabra clave NEWDFT tiene apóstrofos intercalados, éstos deben duplicarse para que el mandato se ejecute correctamente.

```
    CHGJOB PRTTXT('Isn''t this print text')
       .
       .
     CHGCMDDFT CMD(CHGJOB) +
               NEWDFT('PRTTXT(''Isn''''t this print text'')')
```

9.9.1.5 Ejemplo 5

Este ejemplo proporciona el valor por omisión QGPL al primer calificador (nombre de biblioteca) del primer elemento de la lista de la palabra clave DTAMBRS del mandato CRTLF. El nuevo valor por omisión del segundo elemento de la lista de la palabra clave DTAMBRS (nombre de miembro) es MBR1.

```
     CRTLF FILE(FILE1) DTAMBRS(*ALL)
       .
       .
     CHGCMDDFT CMD(CRTLF) +
               NEWDFT('DTAMBRS((QGPL/*N (MBR1)))')
```

Puesto que *ALL es SNGVAL (valor único) para toda la lista DTAMBRS, los valores por omisión de *CURRENT para el nombre de biblioteca y *NONE para el nombre del miembro no aparecen en la pantalla de solicitud de mandatos original. Los valores por omisión *CURRENT y *NONE pueden cambiarse por un valor nuevo, pero no aparecen en la pantalla de solicitud original debido al valor único *ALL para la lista completa DTAMBRS.

9.9.1.6 Ejemplo 6

Crear un mandato que visualizará los archivos de spool para un trabajo:

```
     CRTDUPOBJ OBJ(WRKJOB) FROMLIB(QSYS) +
               TOLIB(MYLIB) NEWOBJ(WRKJOBSPLF)
|    WRKJOBSPLF  OPTION(*SPLF)
       .
       .
     CHGCMDDFT CMD(MYLIB/WRKJOBSPLF) +
               NEWDFT('OPTION(*SPLF)')
```

9.10 Escritura de un Programa o Procedimiento de Proceso de Mandatos

Un programa de proceso de mandatos (CPP) puede ser un programa CL o HLL, o bien un procedimiento REXX. A los programas escritos en CL o HLL también se les puede llamar directamente con el mandato CL CALL. A los procedimientos REXX se les puede llamar directamente mediante el mandato Arrancar Procedimiento REXX (STRREXPRC). No es preciso que exista el programa de proceso de mandatos cuando se ejecuta el mandato Crear Mandato (CRTCMD). Si se utiliza *LIBL como el calificador de bibliotecas, la lista de bibliotecas se utiliza para encontrar el programa de proceso de mandatos cuando se ejecuta el mandato que se ha creado.

Los mensajes emitidos como resultado de la ejecución del programa de proceso de mandatos pueden enviarse a la cola de mensajes del trabajo y visualizarse o imprimirse automáticamente. Puede enviar pantallas a la estación de pantalla solicitante.

Notas:

1. Los parámetros definidos en el mandato se pasan individualmente en el orden en el que se definieron (orden de la sentencia PARM).

2. Los valores decimales se pasan a los programas HLL y CL como valores decimales empaquetados cuya longitud es la especificada en la sentencia PARM.

3. Los valores de caracteres, nombre y los valores lógicos se pasan a los programas HLL y CL como una serie de caracteres de la longitud definida en la sentencia PARM.

Subtemas

9.10.1 Escritura de un Programa de Proceso de Mandatos CL o HLL

La Figura 9-3 muestra la relación entre el mandato Crear Mandato (CRTCMD), las sentencias de definición de mandatos y el programa de proceso de mandatos.

```
CRTCMD CMD(DSPORD) PGM(DSPORDPGM)

            ¦                +--------------------+
            □                                     ¦
  Definición para Mandato DSPORD:                 ¦
  +-----------------------------------+           ¦
  ¦ CMD    'Visualizar Pedidos'       ¦           ¦
  ¦ PARM   KWD (ORDER) +              ¦           ¦
  ¦        TYPE (*DEC) +              ¦           ¦
  ¦        LEN (6 0) +                ¦           ¦
  ¦        RANGE (100000 600000) +    ¦           ¦
  ¦        PROMPT ('Núm pedido   :') ¦           ¦
  +-----------------------------------+           ¦
                                                  □
  Programa Proceso Mandatos para Mandato DSPORD (DSPORDPGM):
                            +-----------------------------------------+
                            ¦ PGM (&ORDER)                            ¦
                            ¦                                         ¦
                            ¦ DCL &ORDER +                            ¦
                            ¦     TYPE (*DEC) +                       ¦
                            ¦     LEN (6 0)                           ¦
                            ¦     RANGE (100000 600000) +             ¦
                            ¦     PROMPT ('Núm. pedido :')            ¦
                            ¦ .                                       ¦
                            ¦ .                                       ¦
                            ¦ .                                       ¦
                            ¦                                         ¦
                            +-----------------------------------------+
```

Figura 9-3. Relaciones entre mandatos para CL y HLL

Si el programa de proceso de mandatos es un programa escrito en CL, las variables que reciben los valores de parámetro deben declararse para que correspondan con el tipo y longitud especificados para cada sentencia PARM. A continuación se muestra esta correspondencia. (Véase la declaración del parámetro ORDER en la Figura 9-3.)

Tipo sentencia PARM	**Long. sentencia PARM**	**Tipo de variable declarada**	**Longitud de variable declarada**
*DEC	x y(1)	*DEC	x y(1)
*LGL	1	*LGL	1
*CHAR	n	*CHAR	=<.n(2)
*NAME	n	*CHAR	=<.n(2)
*CNAME	n	*CHAR	=<.n(2)
*SNAME	n	*CHAR	=<.n(2)
*GENERIC	n	*CHAR	=<.n(2)
*CMDSTR	n	*CHAR	=<.n(2)
*DATE	7	*CHAR	7
*TIME	6	*CHAR	6
*INT2	n	*CHAR	2
*INT4	n	*CHAR	4

(1) x es igual a la longitud e y es el número de posiciones decimales.

(2) Para las variables de tipo carácter, si la longitud del valor pasado es superior a la longitud declarada, el valor se trunca en la longitud declarada. Si se especifica RTNVAL(*YES), la longitud declarada debe ser igual a la longitud definida en la sentencia PARM.

Un programa escrito en CL utilizado como programa de proceso de mandatos puede procesar valores binarios (como *INT2 o *INT4). El programa debe

recibir estos valores como campos de tipo carácter. Puede utilizarse la función incorporada en binario (%BINARY) para convertirlos a valores decimales.

Para ver ejemplo de programas de proceso de mandatos, consulte el apartado "Ejemplos de Definición y Creación de Mandatos" en el tema 9.12.

9.10.2 Escritura de un Procedimiento de Proceso de Mandatos REXX

La Figura 9-4 muestra la relación entre el mandato Crear Mandato (CRTCMD), las sentencias de definición de mandatos y el procedimiento de proceso de mandatos de REXX.

```
CRTCMD CMD(DSPORD) PGM(*REXX) REXSRCMBR(DSPORDPRC)

            ¦                +--------------------+
            □                                     ¦
  Definición para Mandato DSPORD:                 ¦
  +-----------------------------------+           ¦
  ¦ CMD    'Visualizar Pedidos'       ¦           ¦
  ¦ PARM   KWD (ORDER) +              ¦           ¦
  ¦        TYPE (*DEC) +              ¦           ¦
  ¦        LEN (6 0) +                ¦           ¦
  ¦        RANGE (100000 600000) +    ¦           ¦
  ¦        PROMPT ('Núm pedido   :') ¦           ¦
  +-----------------------------------+           ¦
                                                  □
  Procedimiento Proceso Mandatos para Mandato DSPORD (DSPORDPRC):
                           +-----------------------------------------------+
                           ¦ REM Analiz arg . 'ORDER('pedido')'.           ¦
                           ¦ .                                             ¦
                           ¦ .              +------------------+           ¦
                           ¦ .                                             ¦
                           ¦                                               ¦
                           +-----------------------------------------------+
```

Figura 9-4. Relaciones entre mandatos para REXX

9.11 Escritura de un Programa de Comprobación de Validez

Si escribe un programa de comprobación de validez para su mandato, especifique el nombre de dicho programa en el parámetro VLDCKR del mandato Crear Mandato (CRTCMD). El programa no tiene por qué existir cuando se ejecuta el mandato CRTCMD. Si se utiliza *LIBL como el calificador de bibliotecas, la lista de bibliotecas se utiliza para encontrar el programa de comprobación de validez cuando se ejecuta el mandato que se ha creado.

Las siguientes son dos consideraciones acerca de los programas de comprobación de validez:

- Se llama al programa de comprobación de validez solamente si la sintaxis del mandato es correcta. Todos los parámetros se pasan al programa de la misma forma que se pasan a un programa de proceso de mandatos.
- No se debería utilizar el programa de comprobación de validez para modificar los valores de los parámetros, debido a que los valores modificados no siempre se pasan al programa de proceso de mandatos.

El resto de esta sección describe el modo de enviar mensajes desde un programa de comprobación de validez escrito en CL, y reconocido por el sistema como un programa CL.

Si el programa de comprobación de validez detecta un error, debe enviar un mensaje de diagnóstico para la llamada anterior y enviar entonces un mensaje de escape CPF0002. Por ejemplo, si se necesita un mensaje que indique que un número de cuenta ya no es válido, se añade al archivo de mensajes una descripción de mensaje parecida a la que se muestra a continuación:

```
ADDMSGD      MSG('Longitud no válida de número de cuenta &2') +
             MSGID(USR0012) +
             MSGF(QGPL/ACTMSG) +
             SEV(40) +
             FMT((*CHAR 4) (*CHAR 6))
```

Observar que la variable de sustitución &1 no está en el mensaje, pero está definida en el parámetro FMT como de 4 caracteres. &1 está reservada para utilizar por el sistema, y debe ser siempre de 4 caracteres. Si la variable de sustitución &1 es la única variable de sustitución definida en el mensaje, compruebe que el cuarto byte de los datos del mensaje no contienen un blanco cuando envíe el mensaje.

Este mensaje puede enviarse al sistema especificando lo siguiente en la comprobación de validez:

```
SNDPGMMSG    MSGID(USR0012) MSGF(QGPL/ACTMSG) +
             MSGDTA('0000' || &ACCOUNT) MSGTYPE(*DIAG)
```

Cuando la comprobación de validez ha enviado todos los mensajes de diagnóstico necesarios, debe enviar el mensaje CPF0002. El mandato Enviar Mensaje de Programa (SNDPGMMSG) para enviar el mensaje CPF0002 sería el siguiente:

```
SNDPGMMSG    MSGID(CPF0002) MSGF(QCPFMSG) +
             MSGTYPE(*ESCAPE)
```

Cuando el sistema recibe el mensaje CPF0002, envía el mensaje CPF0001 al programa de llamada para indicar que se han hallado errores.

El mensaje CPD0006 ha sido definido para su utilización por los programas de comprobación de validez definidos por el usuario. Puede enviarse un mensaje inmediato en los datos del mensaje. Observe en el ejemplo siguiente que el mensaje debe ir precedido por cuatro caracteres cero.

A continuación se muestra un ejemplo de un programa de comprobación de validez:

```
PGM PARM(&PARM01)
DCL VAR(&PARM01) TYPE(*CHAR) LEN(10)
IF COND(&PARM01 *EQ 'ERROR') THEN(DO)
SNDPGMMSG MSGID(CPD0006) MSGF(QCPFMSG) +
          MSGDTA('0000 MENSAJE DE DIAGNÓSTICO DESDE COMPROBACIÓN DE VALIDEZ +
          DEFINIDA POR USUARIO QUE INDICA QUE PARM01 ES ERRÓNEO.') +
          MSGTYPE(*DIAG)
SNDPGMMSG MSGID(CPF0002) MSGF(QCPFMSG) MSGTYPE(*ESCAPE)
ENDDO
ELSE
 .
 .
 .
ENDPGM
```

9.12 Ejemplos de Definición y Creación de Mandatos

Este apartado contiene ejemplos de definición y creación de mandatos.

Subtemas

9.12.1 Llamadas a Programas de Aplicación

Puede crear mandatos para llamar a los programas de aplicación. Si crea un mandato para llamar a un programa de aplicación, OS/400 efectúa una comprobación de validez sobre los parámetros pasados al programa. Sin embargo, si utiliza el mandato CALL para llamar a un programa de aplicación, es éste el que debe efectuar la comprobación de validez.

Por ejemplo, un programa de escritura de etiquetas (LBLWRT) escribe un número cualquiera de etiquetas para un cliente determinado en formularios compuestos de 1 ó 2 partes. Cuando se ejecuta el programa LBLWRT, se requieren tres parámetros: el número del cliente, la cantidad de etiquetas y el tipo de formulario que se utilizará (ONE o TWO).

Si se llamara al programa directamente desde la pantalla, el segundo parámetro tendría un formato inadecuado para el programa. Una constante numérica en el mandato CALL siempre consta de 15 dígitos con 5 posiciones decimales, y el programa LBLWRT espera un número de 3 dígitos sin posiciones decimales. Puede crearse un mandato que proporcione los datos en el formato que precisa el programa.

Las sentencias de definición de un mandato para llamar el programa LBLWRT son éstas:

```
  CMD PROMPT('Programa de escritura de etiquetas')
  PARM  KWD(CUSNBR) TYPE(*CHAR) LEN(5) MIN(1) +
        PROMPT('Número de cliente')
  PARM  KWD(COUNT) TYPE(*DEC) LEN(3) DFT(20) RANGE(10 150) +
        PROMPT('Cantidad de etiquetas')
  PARM  KWD(FRMTYP) TYPE(*CHAR) LEN(3) DFT('TWO') RSTD(*YES) +
        SPCVAL(('ONE') ('TWO') ('1' 'ONE') ('2' 'TWO')) +
        PROMPT('Tipo de formulario')
```

Para el segundo parámetro, COUNT, se ha especificado un valor por omisión de 20, y el parámetro RANGE sólo permite que se entren valores comprendidos entre 10 y 150 para la cantidad de etiquetas.

Para el tercer parámetro, FRMTYP, el parámetro SPCVAL permite al usuario de la estación de pantalla entrar **'ONE'**, **'TWO'**, **'1'** ó **'2'**. El programa espera el valor **'ONE'** o **'TWO'**; sin embargo, si el usuario de la estación de pantalla entra **'1'** ó '2
para el parámetro FRMTYP.

El programa de proceso de mandatos para este mandato es el programa de
|aplicación LBLWRT. Si el programa de aplicación fuera un programa RPG
|para OS/400, se realizarían las siguientes especificaciones para recibir los parámetros:

```
  *ENTRY   PLIST
           PARM    CUST   5
           PARM    COUNT  30
           PARM    FORM   3
```

El mandato CRTCMD es:

```
  CRTCMD CMD(LBLWRT) PGM(LBLWRT) SRCMBR(LBLWRT)
```

9.12.2 Sustitución de un Valor por Omisión

Puede crear un mandato que proporcione valores por omisión para un mandato suministrado por IBM y que reduzca las entradas que debe realizar el usuario de la estación de pantalla. Por ejemplo, podría crear un mandato Salvar Biblioteca en Cinta (SAVLIBTAP) que inicializara una cinta y salvara una biblioteca en el dispositivo de cinta TAPE1. Este mandato proporciona valores por omisión a los parámetros estándares del mandato Salvar Biblioteca (SAVLIB) y precisa que el usuario de la estación de pantalla especifique solamente el nombre de la biblioteca.

Las sentencias de definición del mandato SAVLIBTAP son las siguientes:

```
  CMD PROMPT('Salvar biblioteca en cinta')
  PARM KWD(LIB) TYPE(*NAME) LEN(10) MIN(1) +
    PROMPT('Nombre de biblioteca')
```

El programa de proceso de mandatos es:

```
  PGM PARM(&LIB)
  DCL &LIB TYPE(*CHAR) LEN(10)
  INZTAP DEV(TAPE1) CHECK(*NO)
  SAVLIB LIB(&LIB) DEV(TAPE1)
  ENDPGM
```

El mandato CRTCMD es:

```
  CRTCMD CMD(SAVLIBTAP) PGM(SAVLIBTAP) SRCMBR(SAVLIBTAP)
```

9.12.3 Visualización de una Cola de Salida

Puede crear un mandato para visualizar una cola de salida cuyo valor por omisión es visualizar la cola de salida PGMR. El mandato siguiente, DSPOQ, también permite al usuario de la estación de pantalla visualizar cualquier cola de la lista de bibliotecas y proporciona una opción de impresión.

Las sentencias de definición del mandato DSPOQ son las siguientes:

```
  CMD PROMPT('WRKOUTQ.-Toma el valor por omisión de PGMR')
  PARM  KWD(OUTQ) TYPE(*NAME) LEN(10) DFT(PGMR) +
        PROMPT('Cola de salida:')
  PARM  KWD(OUTPUT) TYPE(*CHAR) LEN(6) DFT(*) RSTD(*YES)
        VALUES(* *PRINT) PROMPT ('Salida')
```

El parámetro RSTD de la segunda sentencia PARM especifica que la entrada puede ser solamente una que figure en la lista de valores.

El programa de proceso de mandatos para el mandato DSPOQ es:

```
  PGM PARM(&OUTQ &OUTPUT)
  DCL &OUTQ TYPE(*CHAR) LEN(10)
  DCL &OUTPUT TYPE(*CHAR) LEN(6)
  WRKOUTQ OUTQ(*LIBL/&OUTQ) OUTPUT(&OUTPUT)
  ENDPGM
```

El mandato CRTCMD es:

```
  CRTCMD CMD(DSPOQ) PGM(DSPOQ) SRCMBR(DSPOQ)
```

El mandato siguiente, DSPOQ1, es una variación del mandato anterior. Este mandato permite que el usuario de la estación de trabajo entre un nombre calificado para el nombre de la cola de salida; el mandato toma el valor por omisión *LIBL para el nombre de biblioteca.

Las sentencias de definición del mandato DSPOQ1 son las siguientes:

```
            CMD PROMPT('WRKOUTQ.-Toma el valor por omisión de PGMR')
            PARM    KWD(OUTQ) TYPE(QUAL1) +
                    PROMPT('Cola de salida:')
            PARM    KWD(OUTPUT) TYPE(*CHAR) LEN(6) RSTD(*YES) +
                    VALUES(* *PRINT) DFT(*) +
                    PROMPT('Salida')
  QUAL1:    QUAL TYPE(*NAME) LEN(10) DFT(PGMR)
            QUAL TYPE(*NAME) LEN(10) DFT(*LIBL) +
            SPCVAL(*LIBL)
```

Las sentencias QUAL se utilizan para definir el nombre calificado que el usuario puede entrar para el parámetros OUTQ. Si el usuario no entra un nombre, se utiliza *LIBL/PGMR. El parámetro SPCVAL se utiliza porque cualquier nombre de biblioteca debe seguir las normas para un nombre válido (por ejemplo, empezar por una letra de la A a la Z) y el valor *LIBL no cumple estas normas. El parámetro SPCVAL especifica que si se introduce *LIBL, OS/400 pasará por alto las normas de validación de nombres.

El programa de proceso de mandatos para el mandato DSPOQ1 es:

```
  PGM PARM(&OUTQ &OUTPUT)
  DCL &OUTQ TYPE(*CHAR) LEN(20)
  DCL &OBJNAM TYPE(*CHAR) LEN(10)
  DCL &LIB TYPE(*CHAR) LEN(10)
  DCL &OUTPUT TYPE(*CHAR) LEN(6)
  CHGVAR &OBJNAM %SUBSTRING(&OUTQ 1 10)
  CHGVAR &LIB %SUBSTRING(&OUTQ 11 10)
  WRKOUTQ OUTQ(&LIB/&OBJNAM) OUTPUT(&OUTPUT)
  ENDPGM
```

Puesto que un nombre calificado se pasa de un mandato como una variable de 20 caracteres, en este programa debe utilizarse la función incorporada de subseries (%SUBSTRING o %SST) para que la sintaxis del nombre calificado sea correcta para CL.

9.12.4 Visualización de Mensajes de Mandatos IBM Más de Una Vez

El mandato CLROUTQ emite el mensaje de terminación CPF3417, que describe el número de entradas suprimidas, el número de entradas que no se han suprimido y el nombre de la cola de salida. Si el mandato CLROUTQ se ejecuta en un CPP, el mensaje se emite pero se convierte en un mensaje detallado, ya que no lo emite directamente el CPP. Por ejemplo, si un mandato CLROUTQ definido por el usuario se emite desde el Menú del Programador, el mensaje no se visualiza. No obstante, puede recibir un mensaje IBM y volverlo a emitir desde el CPP.

Por ejemplo, puede crear un mandato llamado CQ2 para borrar la cola de salida QPRINT2.

Las sentencias de definición del mandato CQ2 son las siguientes:

```
  CMD PROMPT ('Borrar cola de salida QPRINT2')
```

El mandato CRTCMD es:

```
  CRTCMD CMD(CQ2) PGM(CQ2)
```

El CPP, que recibe el mensaje de terminación y lo visualiza, es el siguiente:

```
  PGM /* Borrar cola salida QPRINT2 CPP */
  DCL &MSGID TYPE(*CHAR) LEN(7)
  DCL &MSGDTA TYPE(*CHAR) LEN(100)
  CLROUTQ QPRINT2
  RCVMSG MSGID(&MSGID) MSGDTA(&MSGDTA) MSGTYPE(*COMP)
  SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGDTA(&MSGDTA) MSGTYPE(*COMP)
  ENDPGM
```

La longitud de MSGDTA para el mensaje CPF3417 es de 28 bytes. Sin embargo, al definir la variable &MSGDTA como de 100 bytes, puede utilizarse el mismo planteamiento en la mayor parte de los mensajes porque se pasan por alto las posiciones que no se utilizan.

9.12.5 Creación de Mandatos Abreviados

Subtemas
9.12.5.1 Ejemplo 1
9.12.5.2 Ejemplo 2

9.12.5.1 Ejemplo 1

Puede crear sus propios mandatos abreviados para simplificar los mandatos proporcionados por IBM, o para restringir los parámetros permitidos a los usuarios. Por ejemplo, para permitir a los usuarios modificar sólo el parámetro de dispositivo de impresora, puede crear su propio mandato Cambiar Trabajo (CJ). A continuación se muestran tres pasos para crear e implementar su propio mandato CJ:

- Paso uno: sentencias fuente de definición de mandatos

```
          CMD   PROMPT('Cambiar Trabajo')

          PARM  KWD(PRTDEV) +
                TYPE(*NAME) +
                LEN(10)  +
                SPCVAL(*SAME *USRPRF *SYSVAL *WRKSTN) +
                PROMPT('Dispositivo Impresora')
```

- Paso dos: Proceso del programa

```
          PGM PARM(&PRTDEV)
          DCL VAR(&PRTDEV)  TYPE(*CHAR)  LEN(10)
          CHGJOB  PRTDEV(&PRTDEV)
          ENDPGM
```

- Paso tres: mandato CRTCMD

```
          CRTCMD  CMD(CJ)  PGM(CJ)  SRCMBR(CJ)
```

9.12.5.2 Ejemplo 2

Puede crear un mandato abreviado denominado DW1 para arrancar el transcriptor de impresora W1.

La sentencia de definición del mandato es:

```
  CMD /* Mandato arrancar transcriptor de impresora */
```

El programa de proceso de mandatos es:

```
  PGM
  STRPRTWTR DEV(QSYSPRT) OUTQ(QPRINT) WTR(W1)
  ENDPGM
```

El mandato CRTCMD es:

```
  CRTCMD CMD(DW1) PGM(DW1) SRCMBR(DW1)
```

9.12.6 Adición o Sustracción de un valor a una Fecha

Puede crear un mandato que sume o reste un número de días especificado por el usuario en una fecha y devuelva la nueva fecha como una variable. Consulte el miembro ADDDAT del archivo QATTINFO en la biblioteca QUSRTOOL para obtener más información.

9.12.7 Supresión de Archivos y Miembros Fuente

Puede crear un mandato para suprimir archivos y sus correspondientes miembros fuente en QDDSSRC.

Las sentencias de definición de un mandato denominado DFS son:

```
  CMD PROMPT('Suprimir archivo y fuente')
  PARM KWD(FILE) TYPE(*NAME) LEN(10) PROMPT('Nombre de archivo')
```

El programa de proceso de mandatos está escrito suponiendo que el nombre del archivo y el miembro del archivo fuente es el mismo. El programa supone también que tanto el archivo como el archivo fuente están en la lista de bibliotecas. Si el programa no puede suprimir el archivo, se envía un mensaje de información y el mandato intenta eliminar el miembro fuente. Si el miembro fuente no existe, se envía un mensaje de escape.

El programa de proceso de mandatos es:

```
  PGM PARM(&FILE)
  DCL &FILE TYPE(*CHAR) LEN(10)
  DCL &MSGID TYPE(*CHAR) LEN(7)
  DCL &MSGDTA TYPE(*CHAR) LEN(80)
  DCL &SRCFILE TYPE(*CHAR) LEN(10)
  MONMSG MSGID(CPF0000) EXEC(GOTO ERROR) /* DETECTAR TODOS */
  DLTF &FILE
  MONMSG MSGID(CPF2105) EXEC(DO) /* NO ENCONTRADO */
  RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)
  SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*INFO) +
        MSGDTA(&MSGDTA)
  GOTO TRYDDS
  ENDDO
  RCVMSG MSGTYPE(*COMP) MSGID(&MSGID) MSGDTA(&MSGDTA)
      /* SUPRIMIR ARCHIVO COMPLET */
  SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*COMP) +
            MSGDTA(&MSGDTA) /* TRY IN QDDSSRC FILE */
  TRYDDS:   CHKOBJ QDDSSRC OBJTYPE(*FILE) MBR(&FILE)
            RMVM QDDSSRC MBR(&FILE)
            CHGVAR &SRCFILE 'QDDSSRC'
            GOTO END
  END:      RCVMSG MSGTYPE(*COMP) MSGID(&MSGID) MSGDTA(&MSGDTA)
                /* ELIMINAR MIEMBRO COMPLET */
            SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*COMP) +
               MSGDTA(&MSGDTA)
            RETURN
  ERROR:    RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)
                /* MENSAJE ESCAPE */
            SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
                 MSGDTA(&MSGDTA)
  ENDPGM
```

9.12.8 Supresión de Objetos de Programa

Puede crear un mandato para suprimir programas HLL y sus miembros fuente correspondientes.

Las sentencias de definición del mandato llamado DPS son:

```
  CMD PROMPT ('Suprimir programa y fuente')
  PARM KWD(PGM) TYPE(*NAME) LEN(10) PROMPT('Nombre de programa')
```

El programa de proceso de mandatos está escrito suponiendo que el nombre del programa y el nombre del miembro de archivo fuente es el mismo y que se han utilizado los archivos fuente proporcionados por IBM. El programa supone también que tanto el programa como el archivo fuente están en la lista de bibliotecas. Si el programa no se puede suprimir, se envía un mensaje informativo y el mandato intenta eliminar el miembro fuente. Si el miembro fuente no existe, se envía un mensaje de escape. El programa de proceso de mandatos es:

```
  PGM PARM(&PGM)
  DCL &PGM TYPE(*CHAR) LEN(10)
  DCL &MSGID TYPE(*CHAR) LEN(7)
  DCL &MSGDTA TYPE(*CHAR) LEN(80)
  DCL &SRCFILE TYPE(*CHAR) LEN(10)
  MONMSG MSGID(CPF0000) EXEC(GOTO ERROR) /* DETECTAR TODOS */
  DLTPGM &PGM
  MONMSG MSGID(CPF2105) EXEC(DO) /* NO ENCONTRADO */
  RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)
  SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*INFO) +
        MSGDTA(&MSGDTA)
  GOTO TRYCL  /* INTENTAR SUPRIMIR MIEMBRO FUENTE */
  ENDDO
  RCVMSG MSGTYPE(*COMP) MSGID(&MSGID) MSGDTA(&MSGDTA)
      /* SUPRIMIR PROGRAMA COMPLET */
  SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*COMP) +
       MSGDTA(&MSGDTA) /* INTENTAR EN QCLSRC */
  TRYCL:   CHKOBJ QCLSRC OBJTYPE(*FILE) MBR(&PGM)
           MONMSG MSGID(CPF9815) EXEC(GOTO TRYRPG) /* NO MIEMBRO CL */
           RMVM QCLSRC MBR(&PGM)
           CHGVAR &SRCFILE 'QCLSRC'
           GOTO END
  TRYRPG:   /* INTENTAR EN ARCHIVO QRPGSRC */
           CHKOBJ QRPGSRC OBJTYPE(*FILE) MBR(&PGM)
           MONMSG MSGID(CPF9815) EXEC(GOTO TRYCBL) /* NO MIEMBRO RPG */
           RMVM QRPGSRC MBR(&PGM)
           CHGVAR &SRCFILE 'QRPGSRC'
           GOTO END
  TRYCBL:   /* INTENTAR EN ARCHIVO QCBLSRC */
           CHKOBJ QCBLSRC OBJTYPE(*FILE) MBR(&PGM)
            /* PERMITIR CPF0000 EN ULTIMO ARCH FUENTE PARA NO ENCONTRADO +
           CONDITION */
           RMVM QCBLSRC MBR(&PGM)
           CHGVAR &SRCFILE 'QCBLSRC'
           GOTO END
  TRYNXT:   /* INSERTAR ARCHIVOS FUENTE ADICIONALES */
            /* AÑADIR MONMSG DESPUÉS DE CHKOBJ EN TRYCBL COMO SE +
              HIZO EN TRYCL Y TRYRPG */
  END:      RCVMSG MSGTYPE(*COMP) MSGID(&MSGID) MSGDTA(&MSGDTA)
                /* ELIMINAR MIEMBRO COMPLET */
            SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*COMP) +
               MSGDTA(&MSGDTA)
            RETURN
  ERROR:    RCVMSG MSGTYPE(*EXCP) MSGID(&MSGID) MSGDTA(&MSGDTA)
                /* MENSAJE ESCAPE */
            SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
                 MSGDTA(&MSGDTA)
  ENDPGM
```

10.0 Capítulo 10. Depuración de Programas ILE

La depuración le permite detectar, diagnosticar y eliminar errores en un programa. Puede depurar sus programas ILE utilizando el depurador de fuente ILE. Este capítulo describe cómo utilizar el depurador del fuente ILE.

Este capítulo describes cómo:

- Preparar su programa ILE para la depuración
- Arrancar una sesión de depuración
- Añadir y eliminar programas de una sesión de depuración
- Visualizar el fuente del programa de una sesión de depuración
- Establecer y eliminar puntos de interrupción condicionales e incondicionales
- Seguir los pasos de un programa
- Visualizar el valor de variables
- Cambiar el valor de variables
- Visualizar los atributos de variables
- Igualar un nombre abreviado a una variable, expresión o mandato de depuración.

Al depurar y probar sus programas, compruebe que la lista de bibliotecas se cambia para dirigir los programas a una biblioteca de prueba que contiene datos de prueba, de modo que no queden afectados los datos reales.

Puede impedir que se modifiquen accidentalmente archivos de base de datos en bibliotecas de producción utilizando uno de los siguientes mandatos:

- Utilice el mandato Arrancar Depuración (STRDBG) y retenga el valor por omisión *NO para el parámetro UPDPROD.
- Utilice el mandato Cambiar Depuración (CHGDBG).

Consulte el manual CL Reference para obtener más información.

Consulte el manual *ILE Conceptos* Capítulo 9, "Consideraciones acerca de la depuración", para obtener más información acerca del depurador del fuente ILE (incluyendo la autorización necesaria para depurar un programa o programa de servicio y los efectos de los niveles de optimización.)

Subtemas

10.1 El Depurador del Fuente ILE

El depurador del fuente ILE se utiliza para detectar errores y eliminar errores de objetos programa y programa de servicio. Puede utilizar el depurador del fuente para:

- Depurar cualquier aplicación de lenguaje CL ILE o ILE mixto
- Supervisar el flujo de un programa utilizando los mandatos de Depuración mientras se ejecuta el programa.
- Visualizar el fuente del programa
- Establecer y eliminar puntos de interrupción condicionales e incondicionales
- Seguir los pasos de un número especificado de sentencias
- Visualizar o cambiar el valor de variables
- Visualizar los atributos de una variable

Cuando un programa se detiene debido a un punto de interrupción o a un mandato de paso, se muestra la vista del objeto módulo aplicable en la pantalla en el punto en que se ha detenido el programa. En este punto puede entrar más mandatos de depuración.

Antes de poder utilizar el depurador del fuente, debe utilizar las opciones de depuración (DBGVIEW) al crear un objeto módulo o programa utilizando el mandato Crear Módulo CL (CRTCLMOD) o Crear CL Enlazado (CRTBNDCL). Una vez establecidos los puntos de interrupción u otras opciones del depurador del fuente ILE, puede llamar al programa.

10.2 Mandatos de Depuración

Hay muchos mandatos de depuración disponibles para utilizarlos con el depurador del fuente ILE. Los mandatos de depuración y sus parámetros se entran en la línea de mandatos de depuración visualizada en la parte inferior de las pantallas Visualizar Fuente de Módulo y Evaluar Expresión. Estos mandatos pueden entrarse en mayúsculas, minúsculas o mixto.

Nota: Los mandatos de depuración entrados en la línea de mandatos del depurador del fuente no son mandatos CL.

La Tabla 10-1 ofrece un resumen de estos mandatos de depuración. La ayuda en línea para el depurador del fuente ILE describe los mandatos de depuración y explica sus abreviaturas permitidas.

Tabla 10-1. Mandatos de depurador del fuente ILE

Mandato de Depuración	**Descripción**
ATTR	Le permite visualizar los atributos de una variable. Los atributos son el tamaño y tipo de la variable tal como se ha registrado en la tabla de símbolos de depuración.
BREAK	Le permite entrar un punto de interrupción condicional o incondicional en una posición del programa que se prueba. Utilice BREAK **posición** WHEN **expresión** para entrar un punto de interrupción condicional.
CLEAR	Le permite eliminar puntos de interrupción condicionales e incondicionales.
DISPLAY	Le permite visualizar los nombres y definiciones asignados al utilizar el mandato EQUATE. También le permite visualizar un módulo de fuente diferente del que se muestra actualmente en la pantalla Visualizar Fuente de Módulo. El objeto módulo debe existir en el objeto programa actual.
EQUATE	Le permite asignar una expresión, variable o mandato de depuración a un nombre para su uso como abreviatura.
EVAL	Le permite visualizar o cambiar el valor de una variable o visualizar el valor de expresiones.
QUAL	Le permite definir el ámbito de las variables que aparecen en mandatos EVAL posteriores.
STEP	Le permite ejecutar una o más sentencias del programa que se depura.
FIND	Busca un número de línea, serie o texto especificado en el módulo que se visualiza actualmente.
UP	Mueve la ventana visualizada del fuente hacia el principio de la vista las posiciones especificadas.
DOWN	Mueve la ventana del fuente visualizada las posiciones especificadas hacia el final de la vista.
LEFT	Mueve la ventana del fuente visualizada hacia la izquierda según el número de caracteres especificado.
RIGHT	Mueve la ventana del fuente visualizada hacia la derecha según el número de caracteres especificado.
TOP	Sitúa la vista para mostrar la primera línea.
BOTTOM	Sitúa la vista para mostrar la última línea.
NEXT	Sitúa la vista en el siguiente punto de interrupción del fuente visualizado actualmente.
PREVIOUS	Sitúa la vista al punto de interrupción anterior del fuente visualizado actualmente.
HELP	Muestra la información de ayuda en línea para los mandatos del depurador del fuente disponibles.
SET	Especifica si la búsqueda es o no sensible a las mayúsculas y minúsculas para todas las subsiguientes peticiones de FIND en la sesión de depuración actual.

		También le permite cambiar el valor de actualización de los archivos de producción.

10.3 Preparación de un Objeto Programa para una Sesión de Depuración

Antes de poder utilizar el depurador del fuente ILE, debe utilizar el mandato CRTCLMOD o CRTBNDCL y especificar la opción DBGVIEW.

Para cada objeto módulo CL ILE que desee depurar, puede crear una de las tres vistas:

- Vista del fuente raíz
- Vista de listado
- Vista de sentencias

Subtemas

10.3.1 Utilización de una Vista de Fuente Raíz

Una vista de fuente raíz contiene las sentencias fuente del miembro fuente.

Para utilizar la vista de fuente raíz con el depurador del fuente ILE, el compilador ILE CL crea la vista de fuente raíz mientras se crea el objeto módulo (*MODULE).

Nota: El objeto módulo se crea utilizando referencias a ubicaciones de las sentencias fuente en el miembro fuente raíz en vez de copiar las sentencias fuente en la vista. Por consiguiente, no modifique, redenomine o mueva miembros fuente raíz entre la creación del módulo y la depuración del módulo creado desde estos miembros.

Para depurar un objeto módulo ILE CL utilizando una vista de fuente raíz, utilice la opción *SOURCE o *ALL en el parámetro DBGVIEW para los mandatos CRTCLMOD o CRTBNDCL.

Una forma de crear una vista de fuente raíz es la siguiente:

```
CRTCLMOD MODULE(MYLIB/MYPGM)
SRCFILE(MYLIB/QCLLESRC) SRCMBR(MYPGM)
TEXT('Programa CL') DBGVIEW(*SOURCE)
```

El mandato Crear Módulo CL (CRTCLMOD) con *SOURCE para el parámetro DBGVIEW crea una vista de fuente raíz para el objeto módulo **MYPGM**.

10.3.2 Utilización de una Vista de Listado

Una vista de listado es similar a la parte de código fuente del listado de compilación o archivo de spool producido por el compilador ILE CL.

Para depurar un objeto módulo ILE CL utilizando una vista de listado, utilice la opción *LIST o *ALL del parámetro DBGVIEW para el mandato CRTCLMOD o CRTBNDCL al crear el módulo.

Una forma de crear una vista de listado consiste en lo siguiente:

```
CRTCLMOD MODULE(MYLIB/MYPGM)
SRCFILE(MYLIB/QCLLESRC) SRCMBR(MYPGM)
TEXT('Programa CL') DBGVIEW(*LIST)
```

10.3.3 Utilización de una Vista de Sentencias

Una vista de sentencias no contiene ningún dato de fuente CL. Sin embargo, pueden añadirse puntos de interrupción utilizando nombres de procedimiento y números de sentencia encontrados en el listado de compilador. Para depurar un objeto módulo ILE CL utilizando una vista de sentencias, necesita una copia del listado de compilador.

Nota: No se muestran datos en la pantalla Visualizar Fuente de Módulo al utilizar una vista de sentencia para depurar un objeto módulo ILE CL.

Para depurar un objeto módulo ILE CL utilizando una vista de sentencias, utilice la opción *STMT, *SOURCE, *LIST o *ALL en el parámetro DBGVIEW para el mandato CRTCLMOD o CRTBNDCL al crear el módulo.

Una forma de crear una vista de sentencias es la siguiente:

```
CRTCLMOD MODULE(MYLIB/MYPGM)
SRCFILE(MYLIB/QLSRC) SRCMBR(MYPGM)
TEXT('Programa CL') DBGVIEW(*STMT)
```

10.4 Arranque del Depurador del Fuente ILE

Una vez creada la vista de depuración, puede empezar a depurar su aplicación.

Para arrancar el depurador del fuente ILE, utilice el mandato Arrancar Depuración (STRDBG). Una vez arrancado el depurador, permanece activo hasta que entra el mandato Finalizar Depuración (ENDDBG).

Inicialmente, puede añadir hasta 10 objetos programa a una sesión de depuración utilizando el parámetro Programa (PGM) del mandato STRDBG. Puede ser cualquier combinación de programas ILE u OPM (modelo de programa original). Para arrancar una sesión de depuración con tres objetos programa, teclee:

```
STRDBG PGM(*LIBL/MYPGM1 *LIBL/MYPGM2 *LIBL/MYPGM3) DBGMODSRC(*YES)
```

Nota: Debe tener autorización *CHANGE sobre un objeto programa para añadirlo a una sesión de depuración.

Tras entrar el mandato STRDBG, aparece la pantalla Visualizar Fuente de Módulo para objetos programa ILE. Se muestra el primer objeto módulo enlazado al objeto programa con datos de depuración.

10.5 Adición de Objetos Programa a una Sesión de Depuración

Puede añadir más objetos programa a una sesión de depuración una vez arrancada la sesión.

Para añadir objetos programa ILE y programas de servicio a una sesión de depuración, utilice la opción 1 (Añadir programa) y teclee el nombre del objeto programa en la primera línea de la pantalla Trabajar con Lista de Módulos. Consulte la Tabla 10-1 en el tema 10.2 para ver una lista de mandatos del depurador del fuente ILE. Puede acceder a la pantalla Trabajar con Lista de Módulos desde la pantalla Visualizar Fuente de Módulo pulsando F14 (Trabajar con Lista de Módulos). Para añadir un programa de servicio, cambie el tipo de programa por omisión de *PGM a *SRVPGM. No hay limitación en el número de objetos programa y programas de servicio ILE que pueden incluirse en una sesión de depuración en un determinado momento.

```
                          Trabajar con Lista de Módulos
                                                              Sistema:   RCHASDIF
  Teclee opciones, pulse Intro.
    1=Añadir programa   4=Eliminar programa   5=Visualizar fuente de módulo
    8=Trabajar con puntos de interrupción de módulo

  Opc     Programa/módulo    Bibliot        Tipo
   1       weekday2          *LIBL          *PGM
           DSPWKDAY          MYLIB          *PGM
             DSPWKDAY                       *MODULE     Seleccionado
             AABP1                          *MODULE

                                                                          Final
  Mandato
  ===>
  F3=Salir  F4=Solicitud  F5=Renovar   F9=Recuperar   F12=Cancelar
```

Figura 10-1. Adición de un Objeto Programa ILE a una Sesión de Depuración. Cuando se pulsa Intro, el programa WEEKDAY2 se añade a la sesión de depuración.

```
                          Trabajar con Lista de Módulos
                                                              Sistema:   RCHASDIF
  Teclee opciones, pulse Intro.
    1=Añadir programa   4=Eliminar programa   5=Visualizar fuente de módulo
    8=Trabajar con puntos de interrupción de módulo

  Opc     Programa/módulo    Bibliot        Tipo
           WEEKDAY2          *LIBL          *PGM
           WEEKDAY2          MYLIB          *PGM
             WEEKDAY2                       *MODULE
           DSPWKDAY          MYLIB          *PGM
             DSPWKDAY                       *MODULE     Seleccionado
             AABP1                          *MODULE

                                                                          Final
 Mandato
  ===>
  F3=Salir  F4=Solicitud  F5=Renovar   F9=Recuperar   F12=Cancelar
  Programa WEEKDAY2 añadido al depurador del fuente.
```

Figura 10-2. Adición de un Objeto Programa ILE a una Sesión de

Depuración. El mensaje informativo en la parte inferior de la pantalla muestra que el programa WEEKDAY2 se ha añadido a la sesión de depuración.

Una vez finalizada la adición de objetos programa a la sesión de depuración, pulse F3 (Salir) desde la pantalla Trabajar con Lista de Módulos para volver a la pantalla Visualizar Fuente de Módulo. También puede utilizar la opción F5 (Renovar) para seleccionar y visualizar un módulo. Cuando pulse F3 desde la pantalla de Renovar, volverá a la pantalla Visualizar Fuente de Módulo.

Nota: Debe tener autorización *CHANGE sobre un programa para añadirlo a una sesión de depuración. Los programas de servicio ILE pueden añadirse a una sesión de depuración solamente mediante la opción 1 de la pantalla Trabajar con Lista de Módulos. No pueden especificarse programas de servicio ILE en el mandato STRDBG.

Para añadir programas OPM a una sesión de depuración, utilice el mandato Añadir Programa (ADDPGM). Sólo se pueden incluir 10 programas OPM en una sesión de depuración en un determinado momento.

10.6 Eliminación de Objetos Programa de una Sesión de Depuración

Puede eliminar objetos programa de una sesión de depuración una vez arrancada la sesión.

Para eliminar objetos programa y programas de servicio ILE de una sesión de depuración, utilice la opción 4 (Eliminar programa) junto al objeto programa que desea eliminar en la pantalla Trabajar con Lista de Módulos. Consulte la Figura 10-3. Puede acceder a la pantalla Trabajar con Lista de Módulos desde la pantalla Visualizar Fuente de Módulo pulsando F14 (Trabajar con Lista de Módulos). Para eliminar un programa de servicio, cambie el tipo de programa por omisión de *PGM a *SRVPGM.

```
                          Trabajar con Lista de Módulos
                                                               Sistema:   RCHASDIF
  Teclee opciones, pulse Intro.
    1=Añadir programa   4=Eliminar programa   5=Visualizar fuente de módulo
    8=Trabajar con puntos de interrupción de módulo

  Opc     Programa/módulo    Bibliot        Tipo
                             *LIBL          *PGM
    4       WEEKDAY2         MYLIB          *PGM
              WEEKDAY2                      *MODULE
            DSPWKDAY         MYLIB          *PGM
              DSPWKDAY                      *MODULE     Seleccionado
              AABP1                         *MODULE









                                                                            Final
 Mandato
  ===>
  F3=Salir  F4=Solicitud  F5=Renovar   F9=Recuperar   F12=Cancelar
```

Figura 10-3. Eliminación de un Objeto Programa ILE de una Sesión de Depuración. Cuando se pulsa Intro, el programa WEEKDAY2 se elimina de la sesión de depuración.

```
                          Trabajar con Lista de Módulos
                                                               Sistema:   RCHASDIF
  Teclee opciones, pulse Intro.
    1=Añadir programa   4=Eliminar programa   5=Visualizar fuente de módulo
    8=Trabajar con puntos de interrupción de módulo

  Opc     Programa/módulo    Bibliot        Tipo
                             *LIBL          *PGM
            DSPWKDAY         MYLIB          *PGM
              DSPWKDAY                      *MODULE     Seleccionado
              AABP1                         *MODULE









                                                                        Final
 Mandato
  ===>
  F3=Salir  F4=Solicitud  F5=Renovar   F9=Recuperar   F12=Cancelar
  Programa WEEKDAY2 eliminado del depurador del fuente.
```

Figura 10-4. Eliminación de un Objeto Programa ILE de una Sesión de Depuración

Cuando haya terminado de eliminar objetos programa de la sesión de depuración, pulse F3 (Salir) desde la pantalla Trabajar con Lista de

Módulos para volver a la pantalla Visualizar Fuente de Módulo.

Nota: Debe tener autorización *CHANGE sobre un programa para eliminarlo de una sesión de depuración.

Para eliminar programas OPM de una sesión de depuración, utilice el mandato Eliminar Programa (RMVPGM).

10.7 Visualización del Fuente del Programa

La pantalla Visualizar Fuente de Módulo muestra el fuente de un objeto programa, mostrando un objeto módulo cada vez. Un fuente de objeto módulo puede visualizarse si el objeto módulo se compiló utilizando una de las siguientes vistas de depuración:

- DBGVIEW(*ALL)
- DBGVIEW(*SOURCE)
- DBGVIEW(*LISTING)

Existen dos métodos de cambiar lo que se visualiza en la pantalla Visualizar Fuente de Módulo:

- Cambiar una vista
- Cambiar un módulo

Cuando cambia una vista, el depurador del fuente ILE establece la correspondencia con las posiciones equivalentes en la vista a la que está cambiando. Cuando cambia el módulo, la sentencia ejecutable de la vista visualizada se almacena en memoria y se visualiza después cuando se vuelve a visualizar el módulo. Se resaltan los números de línea que tienen establecidos puntos de interrupción. Cuando un punto de interrupción, paso o mensaje provoca la parada del programa y la visualización de la pantalla, se resalta la línea del fuente en el que se ha producido el suceso.

10.8 Cambio de un Objeto Módulo

Puede cambiar el objeto módulo que se muestra en la pantalla Visualizar Fuente de Módulo utilizando la opción 5 (Visualizar fuente de módulo) de la pantalla Trabajar con Lista de Módulos. Puede acceder a la pantalla Trabajar con Lista de Módulos desde la pantalla Visualizar Fuente de Módulo pulsando F14 (Trabajar con Lista de Módulos). En la Figura 10-5 se muestra la pantalla Visualizar Fuente de Módulo.

|Para seleccionar un objeto módulo, teclee 5 (Visualizar fuente de módulo)
|junto al objeto módulo que desea visualizar.

```
                          Visualizar Fuente de Módulo

  Programa:  DSPWKDAY       Bibliot:   MYLIB          Módulo:   DSPWKDAY
      24      500-              CALL       PGM(WEEKDAY2) PARM(&DAYOFWK)
      25      600-              IF         COND(&DAYOFWK *EQ 1) THEN(CHGVAR +
      26      700                            VAR(&WEEKDAY) VALUE('Domingo'))
      27      800-              ELSE       CMD(IF COND(&DAYOFWK *EQ 2) THEN(CHGV
      28      900                            VAR(&WEEKDAY) VALUE('Lunes')))
      29     1000-              ELSE       CMD(IF COND(&DAYOFWK *EQ 3) THEN(CHGV
      30     1100                           VAR(&WEEKDAY) VALUE('Martes')))
      31     1200-              ELSE       CMD(IF COND(&DAYOFWK *EQ 4) THEN(CHGV
      32     1300                           VAR(&WEEKDAY) VALUE('Miércoles')))
      33     1400-              ELSE       CMD(IF COND(&DAYOFWK *EQ 5) THEN(CHGV
      34     1500                           VAR(&WEEKDAY) VALUE('Jueves')))
      35     1600-              ELSE       CMD(IF COND(&DAYOFWK *EQ 6) THEN(CHGV
      36     1700                           VAR(&WEEKDAY) VALUE('Viernes')))
      37     1800-              ELSE       CMD(IF COND(&DAYOFWK *EQ 7) THEN(CHGV
      38     1900                           VAR(&WEEKDAY) VALUE('Sábado')))
                                                                        Más...
  Depurar . .

  F5=Renovar   F9=Recuperar  F14=Trab con lista módulos  F15=Seleccionar vista
  F16=Repetir búsq. F19=Izq. F20=Derech.  F22=Ejec. pasos int.  F24=Mas teclas
```

Figura 10-5. Vista de Visualizar un Módulo

Una vez seleccionado el objeto módulo que desea visualizar, pulse Intro. Se muestra el objeto módulo seleccionado en la pantalla Visualizar Fuente de Módulo.

Un método alternativo para cambiar un objeto módulo consiste en utilizar el mandato de depuración DISPLAY. En la línea de mandatos de depuración, teclee:

```
  DISPLAY MODULE nombre-módulo
```

El **nombre-módulo** del objeto módulo no se mostrará. El objeto módulo debe existir en un objeto programa o programa de servicio añadido a la sesión de depuración.

Subtemas

10.8.1 Cambio de Vista de un Objeto Módulo

Hay diversas vistas de un objeto módulo ILE CL disponibles dependiendo de los valores especificados al crear un objeto módulo ILE CL. Estas vistas son:

- Vista del fuente raíz
- Vista de listado
- Vista de sentencias

Puede cambiar la vista del objeto módulo que se muestra en la pantalla Visualizar Fuente de Módulo a través de la pantalla Seleccionar Vista. Se puede acceder a la pantalla Seleccionar Vista desde la pantalla Visualizar Fuente de Módulo pulsando F15 (Seleccionar Vista). La pantalla Seleccionar Vista se muestra en la Figura 10-6. La vista actual se lista en la parte superior de la ventana, y las demás vistas disponibles se muestran a continuación. Cada objeto módulo de un objeto programa puede tener un conjunto diferente de vistas disponibles, según las opciones de depuración utilizadas para crearlo.

Para seleccionar una vista, teclee 1 (Seleccionar) junto a la vista que desea visualizar.

```
                          Visualizar Fuente de Módulo
  ..........................................................................
  :                              Seleccionar Vista                         :
  :                                                                        :
  :  Vista actual . . . :   Fuente Raíz CL                                 :
  :                                                                        :
  :  Teclee opción, pulse Intro.                                           :
  :    1=Seleccionar                                                       :
  :                                                                        :
  :  Opc     Vista                                                         :
  :          Fuente Raíz CL                                                :
  :   1      Vista Listado CL                                              :
  :                                                                        :
  :                                                                        :
  :                                                                Final   :
  :  F12=Cancelar                                                          :
  :                                                                        :
  :........................................................................:
                                                                    Más...
  Depurar . .

  F5=Renovar   F9=Recuperar  F14=Trab con lista módulos  F15=Seleccionar vista
  F16=Repetir búsq. F19=Izq. F20=Derech.  F22=Ejec. pasos int.  F24=Mas teclas
```

Figura 10-6. Cambio de una Vista de un Objeto Módulo

Una vez seleccionada la vista del objeto módulo que desea visualizar, pulse Intro y se mostrará la vista seleccionada del objeto módulo en la pantalla Visualizar Fuente de Módulo.

10.8.2 Establecimiento y Eliminación de Puntos de Interrupción

Puede utilizar puntos de interrupción para detener un objeto programa en un punto específico durante la ejecución. Un **punto de interrupción incondicional** detiene el objeto programa en una sentencia específica. Un **punto de interrupción condicional** detiene el objeto programa cuando se cumple una condición específica.

Cuando el objeto programa se detiene, aparece la pantalla Visualizar Fuente de Módulo. Se muestra el objeto módulo adecuado con el fuente situado en la línea en la que se ha producido el punto de interrupción. Esta línea está resaltada. En este punto, puede evaluar variables, establecer más puntos de interrupción y ejecutar cualquiera de los mandatos de depuración.

Tenga en cuenta las siguientes características de los puntos de interrupción antes de utilizarlos:

- Cuando se salta un punto de interrupción, por ejemplo con la sentencia GOTO, ese punto de interrupción no se procesa.

- Cuando se establece un punto de interrupción en una sentencia, el punto de interrupción se produce antes de que se procese dicha sentencia.

- Cuando se llega a una sentencia con un punto de interrupción condicional, la expresión condicional asociada al punto de interrupción se evalúa antes de que se procese la sentencia.

- Las funciones de punto de interrupción se especifican a través de mandatos de depuración.

  Estas funciones incluyen:

  - Adición de puntos de interrupción a objetos programa
  - Eliminación de puntos de interrupción de objetos programa
  - Visualización de información de punto de interrupción
  - Reanudación de la ejecución de un objeto programa tras llegar a un punto de interrupción.

10.8.3 Establecimiento y Eliminación de Puntos de Interrupción Incondicionales

Puede establecer o eliminar un punto de interrupción incondicional utilizando:

- F6 (Añadir/Borrar punto de interrupción) desde la pantalla Visualizar Fuente de Módulo
- F13 (Trabajar con Puntos de Interrupción de Módulo) desde la pantalla Visualizar Fuente de Módulo
- El mandato de depuración BREAK para establecer un punto de interrupción
- El mandato de depuración CLEAR para eliminar un punto de interrupción

La forma más sencilla de establecer y eliminar un punto de interrupción incondicional consiste en utilizar F6 (Añadir/Borrar punto de interrupción) desde la pantalla Visualizar Fuente de Módulo. Para establecer un punto de interrupción incondicional utilizando F6, coloque el cursor en la línea en la que desea añadir el punto de interrupción y pulse F6. Se establece un punto de interrupción incondicional en la línea. Para eliminar un punto de interrupción incondicional, coloque el cursor en la línea en la que desea eliminar el punto de interrupción y pulse F6. El punto de interrupción se elimina de la línea.

Repita los pasos anteriores para cada punto de interrupción incondicional que desea establecer.

Nota: Si la línea en la que desea establecer un punto de interrupción no es una sentencia ejecutable, el punto de interrupción se establece en la siguiente sentencia ejecutable.

Una vez establecidos los puntos de interrupción, pulse F3 (Salir) para salir de la pantalla Visualizar Fuente de Módulo. También puede utilizar F21 (Línea de mandatos) desde la pantalla Visualizar Fuente de Módulo para llamar al programa desde una línea de mandatos.

Llame al objeto programa. Cuando se llega a un punto de interrupción, el programa se detiene y se vuelve a mostrar la pantalla Visualizar Fuente de Módulo. En este punto, puede evaluar variables, establecer más puntos de interrupción y ejecutar cualquiera de los mandatos de depuración.

Un método alternativo de establecer y eliminar puntos de interrupción incondicionales consiste en utilizar los mandatos de depuración BREAK y CLEAR.

Para establecer un punto de interrupción incondicional utilizando el mandato de depuración BREAK, teclee:

```
BREAK número-línea
```

en la línea de mandatos de depuración. El **número-línea** es el número de línea de la vista visualizada actualmente del objeto módulo en la que desea establecer un punto de interrupción.

Para eliminar un punto de interrupción incondicional utilizando el mandato de depuración CLEAR, teclee:

```
CLEAR número-línea
```

en la línea de mandatos de depuración. El **número-línea** es el número de línea de la vista visualizada actualmente del objeto módulo de la que desea eliminar un punto de interrupción.

Si está utilizando la vista de sentencia, no se visualizan números de línea. Para establecer puntos de interrupción incondicionales en la vista de sentencias, teclee:

```
BREAK nombre-procedimiento/número-sentencia
```

en la línea de mandatos de depuración. El **nombre-procedimiento** es el nombre de su módulo CL. El **Número-sentencia** (del listado de compilador) es el número de sentencia en que desea detenerse.

10.8.4 Establecimiento y Eliminación de Puntos de Interrupción Condicionales

Puede establecer o eliminar un punto de interrupción condicional utilizando:

- La pantalla Trabajar con Puntos de Interrupción
- El mandato de depuración BREAK para establecer un punto de interrupción
- El mandato de depuración CLEAR para eliminar un punto de interrupción

Subtemas
10.8.4.1 Utilización de la Pantalla Trabajar con Puntos de Interrupción
10.8.4.2 Utilización de los Mandatos de Depuración BREAK y CLEAR
10.8.4.3 Secuencia de Ordenación de Idiomas Nacionales (NLSS)
10.8.4.4 Ejemplos de Puntos de Interrupción Condicionales

10.8.4.1 Utilización de la Pantalla Trabajar con Puntos de Interrupción

Nota: Los operadores relacionales soportados para puntos de interrupción condicionales son <, >, =, <=, >=, y <> (no igual).

Una de las formas de establecer o eliminar puntos de interrupción condicionales es mediante la pantalla Trabajar con Puntos de Interrupción de Módulo. Puede acceder a la pantalla Trabajar con Puntos de Interrupción de Módulo desde la pantalla Visualizar Fuente de Módulo pulsando F13 (Trabajar con Puntos de Interrupción de Módulos). En la Figura 10-7 se muestra la pantalla Trabajar con Puntos de Interrupción de Módulo. Para establecer un punto de interrupción condicional, teclee lo siguiente:

- 1 (Añadir) en el campo *Opc*,
- el número de línea de depurador en que desea establecer el punto de interrupción en el campo *Línea*,
- una expresión condicional en el campo *Condición*,

y pulse Intro. Por ejemplo, para establecer un punto de interrupción condicional en la línea 35 del depurador, tal como se muestra en la Figura 10-7, teclee lo siguiente:

- 1 (Añadir) en el campo *Opc*,
- 35 en el campo *Línea*,
- teclee &I=21,2 en el campo *Condición*,

y pulse Intro.

Para eliminar un punto de interrupción condicional, teclee 4 (Borrar) en el campo *Opc* junto al punto de interrupción que desea eliminar y pulse Intro. También puede eliminar puntos de interrupción incondicionales siguiendo este mismo método.

```
                Trabajar con Puntos de Interrupción de Módulo
                                                          Sistema:  RCHASD04
  Programa . . . :   MYPGM               Biblioteca . . :   MYLIB
    Módulo . . . :     MYMOD             Tipo . . . . . :   *PGM

 Teclee opciones, pulse Intro
    1=Añadir  4=Borrar

  Opc     Línea      Condición
   1      35____     &I=21.2_____________________
   _      ______     ____________________________
```

Figura 10-7. Establecimiento de un Punto de Interrupción Condicional

Repita los pasos anteriores para cada punto de interrupción condicional que desee establecer.

Nota: Si la línea en la que desea establecer un punto de interrupción no es una sentencia ejecutable, el punto de interrupción se establece en la siguiente sentencia ejecutable.

Una vez especificados todos los puntos de interrupción que desea establecer o eliminar, pulse F3 (Salir) para volver a la pantalla Visualizar Fuente de Módulo.

A continuación pulse F3 (Salir) para salir de la pantalla Visualizar Fuente de Módulo. También puede utilizar F21 (Línea de mandatos) desde la pantalla Visualizar Fuente de Módulo para llamar al programa desde una línea de mandatos.

Llame al objeto programa. Cuando se llega a una sentencia con un punto de interrupción condicional, la expresión condicional asociada al punto de interrupción se evalúa antes de que se ejecute la sentencia. Si el resultado es falso, el objeto programa sigue ejecutándose. Si el resultado es verdadero, el objeto programa se detiene, y se muestra la pantalla Visualizar Fuente de Módulo. En este punto, puede evaluar variables, establecer más puntos de interrupción y ejecutar cualquiera de los mandatos de depuración.

10.8.4.2 Utilización de los Mandatos de Depuración BREAK y CLEAR

Un método alternativo de establecer y eliminar puntos de interrupción condicionales consiste en utilizar los mandatos de depuración BREAK y CLEAR.

Para establecer un punto de interrupción condicional utilizando el mandato de depuración BREAK, teclee:

```
  BREAK número-línea WHEN expresión
```

en la línea de mandatos de depuración. El **número-línea** es el número de línea de la vista visualizada actualmente del objeto módulo en la que desea establecer un punto de interrupción. **Expresión** es la expresión condicional que se evalúa cuando el punto de interrupción es encontrado. Los operadores relacionales soportados para puntos de interrupción condicionales son <, >, =, <=, >=, y <> (no igual).

En expresiones condicionales no numéricas, la expresión corta se rellena implícitamente con blancos antes de realizarse la comparación. Este relleno implícito se produce antes de cualquier conversión de la Secuencia de Ordenación de Idiomas Nacionales (NLSS). Consulte la sección "Secuencia de Ordenación de Idiomas Nacionales (NLSS)" en el tema 10.8.4.3 para obtener más información referente a NLSS.

Para eliminar un punto de interrupción condicional utilizando el mandato de depuración CLEAR, teclee:

```
  CLEAR número-línea
```

en la línea de mandatos de depuración. El **Número-línea** es un número de la vista que se visualiza actualmente del objeto módulo del que desea eliminar un punto de interrupción.

En la vista de sentencias, no se visualiza ningún número de línea. Para establecer puntos de interrupción condicionales en la vista de sentencias, teclee:

```
| BREAK nombre-procedimiento/nombre-sentencia  expresión
```

en la línea de mandatos de depuración. El **nombre-procedimiento** es el nombre de su módulo CL. El **Número-sentencia** (del listado de compilador) es el número de sentencia en que desea detenerse.

10.8.4.3 Secuencia de Ordenación de Idiomas Nacionales (NLSS)

Las expresiones de punto de interrupción condicional no numéricas se dividen en los dos siguientes tipos:

- 8-Car: cada carácter contiene 8 bits
- 16-Car: cada carácter contiene 16 bits (DBCS)

NLSS sólo se aplica a expresiones de punto de interrupción condicionales no numéricas de tipo 8-Car. Consulte la Tabla 10-2 para ver las combinaciones posibles de expresiones de punto de interrupción condicional no numéricas.

La tabla de secuencia de ordenación utilizada por el depurador del fuente para expresiones de tipo 8-Car, es la tabla de secuencia de ordenación especificada para el parámetro SRTSEQ en los mandatos CRTCLMOD o CRTBNDCL.

Si la tabla de secuencia de ordenación resuelta es *HEX, no se utiliza ninguna tabla de secuencia de ordenación. Por consiguiente, el depurador del fuente utiliza los valores hexadecimales de los caracteres para determinar la secuencia de ordenación. De lo contrario, se utiliza la tabla de secuencia de ordenación especificada para asignar los pesos a cada byte antes de que se realice la comparación. A los bytes entre los caracteres de desplazamiento a teclado ideográfico y a teclado estándar, estos inclusive, **no** se les asignan pesos.

Nota: El nombre de la tabla de secuencia de ordenación se salva durante la compilación. Durante la depuración, el depurador del fuente utiliza el nombre salvado desde la compilación para acceder a la tabla de secuencia de ordenación. Si la tabla de secuencia de ordenación especificada durante la compilación se resuelve en algo distinto de *HEX o *JOBRUN, es importante *no* alterar la tabla de secuencia de ordenación antes de que empiece la depuración. Si no se puede acceder a la tabla porque está dañada o suprimida, el depurador del fuente utiliza la tabla de secuencia de ordenación *HEX.

Tabla 10-2. Expresiones de Puntos de Interrupción Condicionales No Numéricas.

Tipo	**Posibilidades**
8-Car	- Variable de tipo carácter comparada con variable de tipo carácter - Variable de tipo carácter comparada con literal de tipo carácter (1) - Variable de tipo carácter comparada con literal hex (2) - Literal de tipo carácter (1) comparado con variable de tipo carácter - Literal de tipo carácter (1) comparado con literal de tipo carácter (1) - Literal tipo carácter (1) comparado con literal hex (2) - Literal hex (2) comparado con variable de tipo carácter (1) - Literal hex (2) comparado con literal de tipo carácter (1) - Literal hex (2) comparado con literal hex (2)
16-Car	- Variable de tipo carácter DBCS comparada con variable de tipo carácter DBCS - Variable de tipo carácter DBCS comparada con literal gráfico (3) - Variable de tipo carácter DBCS comparada con literal hex (2) - Literal gráfico (3) comparado con variable de tipo carácter DBCS - Literal gráfico (3) comparado con literal Gráfico (3) - Literal gráfico (3) comparado con literal hex (2) - Literal hex (2) comparado con variable de tipo carácter DBCS - Literal hex (2) comparado con literal Gráfico (3)

(1) El literal de tipo carácter tiene el formato 'abc'.

(2) El literal hexadecimal tiene el formato X'dígitos hex'.

(3) El literal gráfico tiene el formato G'<so>DBCS datos<si>'. El carácter de desplazamiento a teclado ideográfico se representa como <so> y el de desplazamiento a teclado estándar se representa como

```
¦   <si>.                                                                    ¦
+-------------------------------------------------------------------------+
```

10.8.4.4 Ejemplos de Puntos de Interrupción Condicionales

```
  Declaraciones CL:  DCL    VAR(&CHAR1) TYPE(*CHAR) LEN(1)
                     DCL    VAR(&CHAR2) TYPE(*CHAR) LEN(2)
                     DCL    VAR(&DEC1) TYPE(*DEC) LEN(3 1)
                     DCL    VAR(&DEC2) TYPE(*DEC) LEN(4 1)


  Mdto depuración:   BREAK 31 WHEN &DEC1 = 48,1

  Mdto depuración:   BREAK 31 WHEN &DEC2 > &DEC1

  Mdto depuración:   BREAK 31 WHEN &CHAR2 <> 'A'

  Comentario:        'A' se rellena implícitamente
                     por la derecha con un blanco antes de
                     realizar la comparación.


  Mdto depuración:   BREAK 31 WHEN %SUBSTR(&CHAR2 2 1) <= X'F1'

  Mdto depuración:   BREAK 31 WHEN %SUBSTR(&CHAR2 1 1) >= &CHAR1

  Mdto depuración:   BREAK 31 WHEN %SUBSTR(&CHAR2 1 1) < %SUBSTR(&CHAR2 2 1)
```

La función incorporada %SUBSTR le permite obtener una subserie de una variable de serie de caracteres. El primer argumento debe ser un identificador de serie, el segundo es la posición inicial y el tercero es el número de caracteres de un solo byte o de doble byte. Los argumentos están delimitados por uno o más espacios.

10.8.5 Eliminación de Todos los Puntos de Interrupción

Puede eliminar todos los puntos de interrupción, condicionales o incondicionales, de un objeto programa que tenga un objeto módulo visualizado en la pantalla Visualizar Fuente de Módulo utilizando el mandato de depuración CLEAR PGM. Para utilizar el mandato de depuración, teclee:

```
  CLEAR PGM
```

en la línea de mandatos de depuración. Los puntos de interrupción se eliminan de todos los módulos enlazados al programa o programa de servicio.

10.9 Seguir los Pasos del Objeto Programa

Tras encontrar un punto de interrupción, puede ejecutar un número especificado de sentencias de un objeto programa, a continuación detener de nuevo el programa y volver a la pantalla Visualizar Fuente de Módulo. El objeto programa empieza a ejecutarse en la siguiente sentencia del objeto módulo en el que se ha detenido el programa. Habitualmente, un punto de interrupción se utiliza para detener el objeto programa.

Puede seguir los pasos de un objeto programa utilizando:

- F10 (Saltar) ó F22 (Entrar en) en la pantalla Visualizar Fuente de Módulo
- El mandato de depuración STEP

Subtemas

10.9.1 Utilización de F10 ó F22 en la pantalla Visualizar Fuente de Módulo

La forma más sencilla de seguir los pasos de un objeto programa sentencia por sentencia consiste en utilizar F10 (Saltar) ó F22 (Entrar en) de la pantalla Visualizar Fuente de Módulo. Cuando pulsa F10 (Saltar) ó F22 (Entrar en), se muestra la siguiente sentencia del objeto módulo mostrado en la pantalla Visualizar Fuente de Módulo, y el objeto programa vuelve a detenerse.

Nota: No puede especificar el número de sentencias para seguir los pasos cuando utiliza F10 (Saltar) ó F22 (Entrar en). Al pulsar F10 (Saltar) ó F22 (Entrar en) se ejecuta un solo paso.

Otro método para seguir los pasos de un objeto programa consiste en utilizar el mandato de depuración STEP. El mandato de depuración STEP le permite ejecutar más de una sentencia en un solo paso.

10.9.2 Utilización del Mandato de Depuración STEP

El número por omisión de sentencias a ejecutar, utilizando el mandato de depuración STEP, es uno. Para seguir los pasos de un objeto programa utilizando el mandato de depuración STEP, teclee:

```
  STEP número-de-sentencias
```

en la línea de mandatos de depuración. El **número-de-sentencias** es el número de sentencias del objeto programa que desea ejecutar en el siguiente paso antes de que se vuelva a detener el programa. Por ejemplo, si teclea

```
  STEP 5
```

En la línea de mandatos de depuración, se ejecutan las siguientes cinco sentencias del objeto programa, después el programa se detiene de nuevo y se muestra la pantalla Visualizar Fuente de Módulo.

10.9.3 Ejecución de Pasos Externos y Ejecución de Pasos Internos

Cuando en una sesión de depuración se encuentra una sentencia CALL a otro objeto programa, puede efectuar una de las siguientes acciones:

- Ejecutar pasos externos del objeto programa llamado, o
- Ejecutar pasos internos del objeto programa llamado.

Si decide **ejecutar pasos externos** del objeto programa llamado, la sentencia CALL y el objeto programa llamado se ejecutan como un solo paso. El objeto programa llamado se ejecuta totalmente antes de que el objeto programa de llamada se detenga en el siguiente paso. La modalidad por omisión es la ejecución de pasos externos.

Si decide **ejecutar pasos internos** del objeto programa llamado, entonces se ejecuta cada sentencia del objeto programa llamado como un solo paso. Si el siguiente paso en el que se va a detener el objeto programa en ejecución está dentro del programa llamado, el objeto programa se detiene en este punto. El objeto programa llamado se muestra en la pantalla Fuente de Módulo si éste es compilado con los datos de depuración, y si el usuario tiene la autorización para depurarlo.

10.10 Ejecución de Pasos Externos en Objetos Programa

Puede ejecutar pasos externos en objetos programa utilizando:

- F10 (Saltar) en la pantalla Visualizar Fuente de Módulo
- El mandato de depuración STEP OVER

Subtemas

10.10.1 Utilización de F10 (Saltar)
10.10.2 Utilización del Mandato de Depuración de Ejecución de Pasos Externos

10.10.1 Utilización de F10 (Saltar)

Puede utilizar F10 (Saltar) en la pantalla Visualizar Fuente de Módulo para ejecutar pasos externos en un objeto programa en una sesión de depuración. Si la siguiente sentencia a ejecutar es una sentencia CALL para otro objeto programa, si pulsa F10 (Saltar) se ejecutará totalmente el objeto programa llamado antes de que se vuelva a detener el objeto programa de llamada.

10.10.2 Utilización del Mandato de Depuración de Ejecución de Pasos Externos

Como alternativa, puede utilizar el mandato de depuración STEP OVER para ejecutar pasos externos en un objeto programa llamado en una sesión de depuración. Para utilizar el mandato de depuración STEP OVER, teclee:

```
STEP número-de-sentencias OVER
```

en la línea de mandatos de depuración. El **número-de-sentencias** es el número de sentencias del objeto programa que desea ejecutar en el siguiente paso antes de que se vuelva a detener el programa. Si una de las sentencias que se ejecutan contiene una sentencia CALL que llama otro objeto programa, el depurador del fuente ILE ejecuta pasos externos del objeto programa llamado.

10.11 Ejecución de Pasos Internos en Objetos Programa

Puede ejecutar pasos internos en objetos programa utilizando:

- F22 (Entrar en) en la pantalla Visualizar Fuente de Módulo
- El mandato de depuración STEP INTO

Subtemas
10.11.1 Utilización de F22 (Entrar en)
10.11.2 Utilización del Mandato de Depuración de Ejecución de Pasos Internos

10.11.1 Utilización de F22 (Entrar en)

Puede utilizar F22 (Entrar en) en la pantalla Visualizar Fuente de Módulo para ejecutar pasos internos en un objeto programa llamado en una sesión de depuración. Si la siguiente sentencia a ejecutar es una sentencia CALL
|que llama a otro objeto programa, entonces pulsar F22 (Entrar en) provoca
|la ejecución de la primera sentencia en el objeto programa llamado.
|Entonces el objeto programa llamado se muestra en la pantalla Visualizar Fuente de Módulo.

Nota: El objeto programa llamado debe tener datos de depuración asociados con el mismo para que se visualice en la pantalla Visualizar Fuente de Módulo.

10.11.2 Utilización del Mandato de Depuración de Ejecución de Pasos Internos

Como alternativa, puede utilizar el mandato de depuración STEP INTO para ejecutar pasos internos en un objeto programa llamado en una sesión de depuración. Para utilizar el mandato de depuración STEP INTO, teclee :

```
STEP número-de-sentencias INTO
```

en la línea de mandatos de depuración. El **número-de-sentencias** es el número de sentencias del objeto programa que desea ejecutar en el siguiente paso antes de que se vuelva a detener el programa. Si una de las sentencias que se ejecutan contiene una sentencia CALL que llama a otro objeto programa, el depurador ejecuta pasos internos en el objeto programa llamado. El objeto programa llamado se ejecuta sentencia a sentencia. Si la ejecución finaliza en el objeto programa llamado, el objeto programa llamado se muestra en la pantalla Visualizar Fuente de Módulo. Por ejemplo, si teclea

```
STEP 5 INTO
```

en la línea de mandatos de depuración, se ejecutan las cinco siguientes sentencias del objeto programa. Si la tercera sentencia es una sentencia CALL que llama a otro objeto programa, entonces se ejecutan dos sentencias del objeto programa de llamada y las tres primeras sentencias del objeto programa llamado.

10.12 Visualización de Variables

Puede visualizar el valor de variables utilizando:

- F11 (Visualizar variable) en la pantalla Visualizar Fuente de Módulo
- El mandato de depuración EVAL

El ámbito de las variables utilizadas en el mandato EVAL se define utilizando el mandato QUAL. Sin embargo, no es necesario definir específicamente el ámbito de las variables de un módulo CL, ya que todas tienen un ámbito global.

Subtemas

10.12.1 Utilización de F11 (Visualizar Variable)

para visualizar una variable utilizando F11 (Visualizar variable), coloque el cursor en la variable que desea visualizar y pulse F11. Se muestra el valor actual de la variable en la línea de mensajes en la parte inferior de la pantalla Visualizar Fuente de Módulo.

```
                          Visualizar Fuente de Módulo

  Programa:  DSPWKDAY       Bibliot:   MYLIB          Módulo:   DSPWKDAY
       4               DCL        VAR(&MSGTEXT) TYPE(*CHAR) LEN(20)
       5               CALL       PGM(WEEKDAY2) PARM(&DAYOFWK)
       6               IF         COND(&DAYOFWK *EQ 1) THEN(CHGVAR +
       7                            VAR(&WEEKDAY) VALUE('Domingo'))
       8               ELSE       CMD(IF COND(&DAYOFWK *EQ 2) THEN(CHGVAR +
       9                            VAR(&WEEKDAY) VALUE('Lunes')))
      10               ELSE       CMD(IF COND(&DAYOFWK *EQ 3) THEN(CHGVAR +
      11                            VAR(&WEEKDAY) VALUE('Martes')))
      12               ELSE       CMD(IF COND(&DAYOFWK *EQ 4) THEN(CHGVAR +
      13                            VAR(&WEEKDAY) VALUE('Miércoles')))
      14               ELSE       CMD(IF COND(&DAYOFWK *EQ 5) THEN(CHGVAR +
      15                            VAR(&WEEKDAY) VALUE('Jueves')))
      16               ELSE       CMD(IF COND(&DAYOFWK *EQ 6) THEN(CHGVAR +
      17                            VAR(&WEEKDAY) VALUE('Viernes')))
      18               ELSE       CMD(IF COND(&DAYOFWK *EQ 7) THEN(CHGVAR +
                                                                      Más...
  Depurar . .

  F3=Final programa F6=Añadir/Borrar punto interrup F10=Ejecutar pasos externos
  F11=Visual var F12=Reanudar F13=Trab puntos interrupción módulo  F24=Más teclas
  &DAYOFWK = 3.
```

Figura 10-8. Visualización de una Variable utilizando F11 (Visualizar variable). Utilización del Mandato de Depuración EVAL

También puede utilizar el mandato de depuración EVAL para determinar el valor de una variable. Para visualizar el valor de una variable utilizando el mandato de depuración EVAL, teclee:

```
  EVAL nombre-variable
```

en la línea de mandatos de depuración. El **nombre-variable** es el nombre de la variable que desea visualizar. El valor de la variable se muestra en la línea de mensajes si se entra el mandato EVAL desde la pantalla Visualizar Fuente de Módulo y el valor puede visualizarse en una sola línea. Si el valor no puede visualizarse en una sola línea, se muestra en la pantalla Evaluar Expresión.

Por ejemplo, para visualizar el valor de la variable **&DAYOFWK** de la línea 7 del objeto módulo mostrado en la Figura 10-8, teclee:

```
  EVAL &DAYOFWK
```

La línea de mensajes de la pantalla Visualizar Fuente de Módulo muestra **&DAYOFWK = 3.** tal como se puede observar en la Figura 10-8.

10.12.2 Ejemplo de visualización de variables lógicas

```
Declaraciones CL:   DCL    VAR(&LGL1) TYPE(*LGL) VALUE('1')

Mdto depuración:    EVAL &LGL1

Resultado:          &LGL1 = '1'
```

10.12.3 Ejemplos de visualización de variables de tipo carácter

```
  Declaraciones CL:   DCL    VAR(&CHAR1) TYPE(*CHAR) LEN(10) VALUE('EXAMPLE')

  Mdto depuración:   EVAL &CHAR1

  Resultado:         &CHAR1 = 'EXAMPLE   '


  Mdto depuración:   EVAL %SUBSTR(&CHAR1 5 3)

  Resultado:         %SUBSTR(&CHAR1 5 3) = 'PLE'


  Mdto depuración:   EVAL %SUBSTR(&CHAR1 7 4)

  Resultado:         %SUBSTR(&CHAR1 5 3) = 'E   '
```

La función incorporada %SUBSTR le permite obtener una subserie de una variable de serie de caracteres. El primer argumento debe ser un identificador de serie, el segundo es la posición inicial y el tercero es el número de caracteres de un solo byte o de doble byte. Los argumentos están delimitados por uno o más espacios.

10.12.4 Ejemplo de visualización de variable decimal

```
Declaraciones CL:  DCL    VAR(&DEC1) TYPE(*DEC) LEN(4 1) VALUE(73.1)

Declaraciones CL:  DCL    VAR(&DEC2) TYPE(*DEC) LEN(3 1) VALUE(12.5)

Mdto depuración:   EVAL &DEC1

Resultado:         &DEC1 = 073,1

Mdto depuración:   EVAL &DEC2

Resultado:         &DEC2 = 12,5
```

10.12.5 Visualización de Variables como Valores Hexadecimales

Puede utilizar el mandato de depuración EVAL para visualizar el valor de variables en formato hexadecimal. Para visualizar una variable en formato hexadecimal, teclee lo siguiente en la línea de mandatos de depuración:

```
EVAL nombre-variable: x  número-de-bytes
```

El **nombre-variable** es el nombre de la variable que desea visualizar en formato hexadecimal. 'x' especifica que la variable se visualizará en formato hexadecimal y el **número-de-bytes** indica el número de bytes visualizado. Si no se especifica la longitud después de 'x', se utiliza el tamaño de la variable como longitud. Siempre se visualizan 16 bytes como mínimo. Si la longitud de la variable es menor que 16 bytes, el espacio restante SE RELLENA CON CEROS hasta llegar al límite de 16 bytes.

```
Declaraciones CL:  DCL    VAR(&CHAR1) TYPE(*CHAR) LEN(10) VALUE('ABC')
                   DCL    VAR(&CHAR2) TYPE(*CHAR) LEN(10) VALUE('DEF')

Mdto depuración:  EVAL &CHAR1:X 32

Resultado:
00000     C1C2C340 40404040 4040C4C5 C6404040 ABC       DEF
00010     40404040 00000000 00000000 00000000     ............
```

10.13 Cambio del Valor de Variables

Puede cambiar el valor de las variables utilizando el mandato EVAL con el operador de asignación (=).

El ámbito de las variables utilizado en el mandato EVAL se define utilizando el mandato QUAL. Sin embargo, no es necesario definir específicamente el ámbito de las variables de un módulo CL, ya que todas tienen un ámbito global.

Puede utilizar el mandato de depuración EVAL para asignar datos numéricos, de tipo carácter o hexadecimales a variables siempre y cuando correspondan con la definición de la variable.

Para cambiar el valor de la variable, teclee:

```
EVAL nombre-variable = valor
```

en la línea de mandatos de depuración. El **nombre-variable** es el nombre de la variable que desea cambiar y el **valor** es un identificador o valor literal que desea asignar a **nombre-variable**. Por ejemplo,

```
EVAL &COUNTER = 3,0
```

cambia el valor de **&COUNTER** a 3,0 y muestra

```
&COUNTER = 3.0 = 3.0
```

en la línea de mensajes de la pantalla Visualizar Fuente de Módulo. El resultado va precedido del nombre-variable y valor que está cambiando.

Cuando asigna valores a una variable de tipo carácter, se aplican las siguientes normas:

- Si la longitud de la expresión fuente es menor que la longitud de la expresión destino, los datos se justifican a la izquierda en la expresión destino y las posiciones restantes se rellenan con blancos.
- Si la longitud de la expresión fuente es mayor que la expresión destino, los datos se justifican a la izquierda en la expresión destino y se truncan.

Nota: Puede asignarse a variables DBCS cualquiera de los siguientes:

- Otra variable DBCS
- Un literal gráfico de tipo G'<so>datos DBCS<si>'
- Un literal hexadecimal de tipo X'dígitos hex'

Subtemas

10.13.1 Ejemplo de cambio de variables lógicas

```
Declaraciones CL:  DCL    VAR(&LGL1) TYPE(*LGL) VALUE('1')
                   DCL    VAR(&LGL2) TYPE(*LGL)


Mdto depuración:   EVAL &LGL1

Resultado:         &LGL1 = '1'


Mdto depuración:   EVAL &LGL1 = X'F0'

Resultado:         &LGL1 = X'F0' = '0'


Mdto depuración:   EVAL &LGL2 = &LGL1

Resultado:         &LGL2 = &LGL1 = '0'
```

10.13.2 Ejemplos de cambiar variables de tipo carácter

```
  Declaraciones CL:  DCL    VAR(&CHAR1) TYPE(*CHAR) LEN(1) VALUE('A')
                     DCL    VAR(&CHAR2) TYPE(*CHAR) LEN(10)


  Mdto depuración:   EVAL &CHAR1 = 'B'

  Resultado:         &CHAR1 = 'B' = 'B'


  Mdto depuración:   EVAL &CHAR1 = X'F0F1F2F3'

  Resultado:         &CHAR1 = 'F0F1F2F3' = '0'


  Mdto depuración:   EVAL &CHAR2 = 'ABC'

  Resultado:         &CHAR2 = 'ABC' = 'ABC       '


  Mdto depuración:   EVAL %SUBSTR(CHAR2 1 2) = %SUBSTR(&CHAR2 3 1)

  Resultado:         %SUBSTR(CHAR2 1 2) = %SUBSTR(&CHAR2 3 1) = 'C '

  Comentario:        Variable &CHAR contiene 'C C       '
```

La función incorporada %SUBSTR le permite obtener una subserie de una variable de serie de caracteres. El primer argumento debe ser un identificador de serie, el segundo es la posición inicial y el tercero es el número de caracteres de un solo byte o de doble byte. Los argumentos están delimitados por uno o más espacios.

10.13.3 Ejemplos de cambiar variables decimales

```
Declaraciones CL:  DCL    VAR(&DEC1) TYPE(*DEC) LEN(3 1) VALUE(73.1)
                   DCL    VAR(&DEC2) TYPE(*DEC) LEN(2 1) VALUE(3.1)


Mdto depuración:   EVAL &DEC1 = 12,3

Resultado:         &DEC1 = 12,3 = 12,3


Mdto depuración:   EVAL &DEC1 = &DEC2

Resultado:         &DEC1 = &DEC2 = 03,1
```

10.14 Ejemplos de Atributos de una Variable

El mandato de depuración Atributo (ATTR) le permite visualizar los
|atributos de una variable. Los atributos son el tamaño (en bytes) y el
tipo de variable tal como se registró en la tabla de símbolos de
depuración en la Tabla 10-1 en el tema 10.2.

A continuación se presenta un ejemplo de utilización del mandato de
depuración ATTR.

```
  Declaración CL:    DCL    VAR(&CHAR2) TYPE(*CHAR) LEN(10)

  Mdto depuración:   ATTR &CHAR2

  Resultado:         TYPE = FIXED LENGTH STRING, LENGTH = 10 BYTES


  Declaración CL:    DCL    VAR(&DEC) TYPE(*DEC) LEN(3 1)

  Mdto depuración:   ATTR &DEC

  Resultado:         TYPE = PACKED(3,1), LENGTH = 2 BYTES
```

10.15 Igualar un Nombre con una Variable, Expresión o Mandato

Puede utilizar el mandato de depuración EQUATE para igualar un nombre con una variable, expresión o mandato de depuración para su uso abreviado. Entonces puede utilizar ese nombre solo o dentro de otra expresión. Si lo utiliza dentro de otra expresión, el valor del nombre se determina antes de que se evalúe la expresión. Estos nombres permanecen activos hasta que finaliza la sesión de depuración o hasta que se elimina un nombre.

Para igualar un nombre con una variable, expresión o mandato de depuración, teclee:

```
  EQUATE definición nombre-abreviado
```

en la línea de mandatos de depuración. El **nombre-abreviado** es el nombre que desea igualar a una variable, expresión o mandato de depuración, y la **definición** es la variable, expresión o mandato de depuración que equipara con el nombre.

Por ejemplo, para definir un nombre abreviado llamado **DC** que visualiza el contenido de una variable llamada **&COUNTER**, teclee:

```
  EQUATE DC EVAL &COUNTER
```

en la línea de mandatos de depuración. Ahora, cada vez que teclee **DC** en la línea de mandatos de depuración, se ejecutará el mandato EVAL **&COUNTER**.

El número máximo de caracteres que pueden teclearse en un mandato EQUATE es de 144. Si no se proporciona una definición y un mandato EQUATE anterior definió el nombre, se elimina la definición anterior. Si el nombre no se había definido anteriormente, se muestra un mensaje de error.

Para ver los nombres que se han definido con el mandato de depuración EQUATE para una sesión de depuración, teclee:

```
  DISPLAY EQUATE
```

en la línea de mandatos de depuración. Se muestra una lista de los nombres activos en la pantalla Evaluar Expresión.

10.16 Soporte de Idiomas Nacionales en Depuración de Fuentes para ILE CL

Cuando trabaja con el Soporte de Idiomas Nacionales en depuración de fuentes para ILE CL se aplican las siguientes condiciones:

- Cuando se muestra una vista en la pantalla Visualizar Fuente de Módulo, el depurador del fuente convierte todos los datos al Identificador de Juego de Caracteres (CCSID) del trabajo de depuración.

- Cuando se asignan literales a variables, el depurador del fuente no ejecuta la conversión a CCSID de los literales entre apóstrofos (por ejemplo 'abc'). Además, los literales entre apóstrofos son sensibles a las mayúsculas y minúsculas.

Subtemas

*10.16.1 Trabajar con la Vista *SOURCE*

La siguiente condición se aplica sólo cuando trabaja con la Vista del Fuente Raíz CL:

- Si el CCSID del archivo fuente es **diferente** del CCSID del módulo, puede que el depurador del fuente no reconozca un identificador CL que contenga caracteres especiales (#, @, $)

El CCSID del módulo puede encontrarse utilizando el mandato Visualizar Módulo (DSPMOD). Si necesita trabajar con una Vista de Fuente Raíz CL y el CCSID del archivo fuente es diferente del CCSID del módulo, puede realizar una de las siguientes acciones:

- Asegurarse de que el CCSID del fuente CL es el mismo que el CCSID del trabajo en tiempo de ejecución.
- Cambiar el CCSID del trabajo en tiempo de ejecución a 65 535 y compilar.
- Utilizar la Vista de Listado CL si las dos opciones anteriores no son posibles.

Consulte el manual *ILE Conceptos*, Capítulo 9, "Consideraciones acerca de la Depuración" para obtener más información sobre las restricciones NLS para Depuración.

10.16.2 Utilización de COPY, SAVE, RESTORE, CRTDUPOBJ y CHKOBJITG durante la Depuración

Los puntos de interrupción o los pasos pueden **suprimirse temporalmente** de un programa durante la depuración, cuando el usuario utiliza ciertos mandatos de lenguaje de control (CL) para especificar su biblioteca o programa. Los puntos de interrupción y los pasos se *restauran* cuando el mandato CL finaliza su ejecución. Aparecerá un mensaje CPD190A en las anotaciones de trabajo cuando se supriman los puntos de interrupción o los pasos, y otro cuando se restauren.

Los siguientes mandatos CL son los que pueden causar una supresión temporal de los puntos de interrupción o de los pasos:

CHKOBJITG	CPY	CPROBJ	RSTLIB	SAVLIB
	CPYLIB	CRTDUPOBJ	RSTOBJ	SAVOBJ
				SAVSYS
				SAVCHGOBJ

Nota: Cuando los mandatos CL operen en el programa, recibirá un mensaje de error CPF7102 cuando emita el mandato BREAK o STEP.

A.0 Apéndice A. Depuración de Programas OPM

Las funciones de prueba están diseñadas para ayudarle a escribir y mantener las aplicaciones. Le permiten ejecutar los programas en un entorno de prueba especial al tiempo que se observa y se controla de cerca el proceso de estos programas en el entorno de prueba. Puede interactuar con estos programas utilizando la funciones de prueba descritas en este capítulo. Estas funciones están disponibles a través de un conjunto de mandatos que pueden utilizarse interactivamente o en un trabajo por lotes. Estas funciones le permiten:

- Rastrear una secuencia de proceso de un programa y mostrar las sentencias procesadas y los valores de las variables de programa en cada punto de la secuencia.

- Detenerse en cualquier sentencia de un programa (denominado punto de interrupción) y recibir el control para realizar una función tal como visualizar o cambiar un valor de variable o llamar a otro programa definido por el usuario.

Dentro del programa que está comprobándose no hay ningún mandato especial específico para la prueba. El mismo programa que está probándose puede ejecutarse normalmente sin cambios. Todos los mandatos de prueba se especifican dentro del trabajo donde está el programa, no como parte permanente del programa que está probándose. Con los mandatos de prueba, se interactúa con los programas simbólicamente y en los mismos términos con los que se escribió el programa de lenguaje de alto nivel (HLL). Se hace referencia a las variables por sus nombres y a las sentencias por sus números. (Son los mismos números utilizados en la lista fuente del programa.) Además, la comprobación de funciones es aplicable solamente al trabajo en el que se estableció. El mismo programa puede utilizarse al mismo tiempo en otro trabajo sin que se vea afectado por las funciones de prueba establecidas.

Subtemas

A.1 Modalidad de Depuración

Para empezar a probar, su programa debe estar en modalidad de depuración. La modalidad de depuración es un entorno especial en el cual se pueden utilizar funciones de prueba además de las funciones normales de sistema. Las funciones de prueba no pueden utilizarse fuera de la modalidad de depuración. Para iniciar la modalidad de depuración, debe usar el mandato Iniciar Depuración (STRDBG). Además de poner el programa en modalidad de depuración, el mandato STRDBG le permite especificar cierta información de comprobación como son los programas que se están depurando. Su programa permanece en modalidad de depuración hasta que se encuentra un mandato Finalizar Depuración (ENDDBG) o Eliminar Programa (RMVPGM) o se acaba el paso de direccionamiento actual.

El mandato STRDBG siguiente coloca el trabajo en modalidad de depuración y añade el programa CUS310 como el programa que se va a depurar.

```
  STRDBG  PGM(CUS310)
```

Subtemas
A.1.1 Adición de Programas a Modalidad de Depuración
A.1.2 Impedir Actualizaciones de Archivos de Bases de Datos en Bibliotecas de Producción

A.1.1 Adición de Programas a Modalidad de Depuración

En la modalidad de depuración se puede ejecutar cualquier programa, pero antes ha de colocarlo en modalidad de depuración. Puede colocar un programa en modalidad de depuración especificándolo en el parámetro PGM en el mandato STRDBG o añadiéndolo a la sesión de depuración con un mandato Añadir Programa (ADDPGM). Puede especificar un máximo de 10 programas para depurarlos simultáneamente en un trabajo. Debe poseer autorización *CHANGE para añadir un programa a la modalidad de depuración.

Si ha especificado 10 programas para la modalidad de depuración (utilizando el mandato STRDBG, ADDPGM o ambos) y desea añadir más programas al trabajo de depuración, debe eliminar algunos de los programas especificados con anterioridad. Para ello, utilice el mandato Eliminar Programa (RMVPGM). Cuando finaliza la modalidad de depuración, todos los programas se eliminan automáticamente de la modalidad de depuración.

Cuando se inicia la modalidad de depuración, puede especificar que un programa sea un un programa por omisión. Mediante la especificación de un programa por omisión, puede utilizar cualquier mandato de depuración que tenga el parámetro PGM sin necesidad de especificar un nombre de programa cada vez que se utiliza un mandato. Esto resulta útil solamente si se está depurando un único programa. Por ejemplo, en el mandato Añadir Punto de Interrupción (ADDBKP), no especificaría un nombre de programa en el parámetro PGM porque se supone que el programa por omisión es el programa en el que se añadirá el punto de interrupción. El nombre de programa por omisión debe estar especificado en la lista de programas a depurar (parámetro PGM). Si en esta lista hay más de un programa, puede especificar el programa por omisión en el parámetro DFTPGM. Si no lo hace, se supone que el primer programa de la lista del parámetro PGM del mandato STRDBG es el programa por omisión.

El programa por omisión se puede cambiar en cualquier momento durante la prueba utilizando el mandato CHGDBG (Cambiar Depuración) o ADDPGM.

Nota: Si un programa que está en modalidad de depuración se elimina, se vuelve a crear o se salva con almacenamiento liberado, las referencias realizadas a este programa (excepto un mandato RMVPGM) pueden acabar en un error de función. Debe eliminar el programa utilizando un mandato RMVPGM o finalizar la modalidad de depuración utilizando un mandato ENDDBG. Si desea cambiar el programa y entonces depurarlo, debe eliminarlo de la modalidad de depuración y después de que se haya creado de nuevo, añadirlo a la modalidad de depuración (mandato ADDPGM).

A.1.2 Impedir Actualizaciones de Archivos de Bases de Datos en Bibliotecas de Producción

Puede utilizar archivos en bibliotecas de producción mientras está en modalidad de depuración. Para evitar que los archivos de base de datos de las bibliotecas de producción se modifiquen involuntariamente, puede especificar UPDPROD(*NO) o el valor por omisión *NO en el mandato STRDBG. De este modo, solamente los archivos de las bibliotecas de prueba se pueden abrir para efectuar actualizaciones o añadir nuevos registros. Si desea abrir archivos de bases de datos de las bibliotecas de producción con este fin, o si desea suprimir miembros de archivos físicos de producción, puede especificar UPDPROD(*YES).

Puede utilizar esta función con la lista de bibliotecas. En la lista de bibliotecas correspondiente al trabajo de depuración, puede colocar una biblioteca de prueba delante de una biblioteca de producción. Es preciso disponer de copias de los archivos de producción que puedan ser actualizados por el programa que se está depurando en la biblioteca de prueba. Así, cuando el programa se ejecuta, utiliza los archivos de la biblioteca de prueba. Por consiguiente, los archivos de producción no pueden actualizarse accidentalmente.

A.2 La Pila de Llamadas

Puede utilizar el mandato Visualizar Depuración (DSPDBG) para visualizar la pila de llamadas, que indica:

- Qué programas se están depurando en un momento determinado
- El número de la instrucción de llamada o el número de instrucción de cada punto de interrupción en el que se detiene el proceso del programa
- El nivel de recurrencia del programa
- Los nombres de los programas que están en modalidad de depuración pero que no se han llamado

Una llamada de un programa es la asignación de almacenamiento *automático* al programa y la transferencia de proceso de máquina al programa. Una serie de llamadas se coloca en una pila de llamadas. Cuando un programa termina el proceso o transfiere el control, se elimina de la pila de llamadas. Para obtener más información acerca de la pila de llamadas, consulte el Capítulo 3.

Se puede llamar a un programa varias veces mientras la primera llamada aún está en la pila de llamadas. Cada llamada de un programa es un nivel de recurrencia del programa.

Cuando finaliza una llamada (el programa vuelve o transfiere el control), se devuelve el almacenamiento automático al sistema.

Notas:

1. Los programas CL pueden ser recurrentes; es decir, un programa CL puede llamarse a sí mismo directa o indirectamente a través de un programa al que ha llamado.

2. Algunos lenguajes de alto nivel no permiten llamadas recurrentes de programas. Otros lenguajes permiten no solamente que los programas sean recurrentes, sino también que lo sean los procedimientos dentro de un programa. (En este manual, el término *nivel de recurrencia* hace referencia al número de veces que el programa se ha llamado en la pila de programas. Se hace referencia explícita a un nivel de recurrencia del procedimiento como el nivel de recurrencia del procedimiento.)

3. Todos los mandatos CL y pantallas utilizan solamente el nivel de recurrencia del nombre calificado del programa.

Subtemas
A.2.1 Activaciones de Programa

A.2.1 Activaciones de Programa

Una activación de un programa es la asignación de almacenamiento estático para el programa. Una activación siempre termina cuando sucede una de las siguientes cosas:

- Finaliza el paso de direccionamiento actual.
- Se cancela la petición que ha activado el programa.
- Se ejecuta el mandato Reclamar Recursos (RCLRSC) de tal forma que finaliza la última (o única) llamada de un programa.

Además, las acciones tomadas durante una llamada de programa pueden destruir una activación. Estas acciones dependen del lenguaje (HLL o CL) en el que esté escrito el programa.

Cuando se desactiva un programa, el almacenamiento estático se devuelve al sistema. El lenguaje (HLL o CL) en el que está escrito el programa determina cuándo se desactiva el programa normalmente. Un programa CL siempre se desactiva cuando finaliza.

Un programa RPG/400 se desactiva cuando se activa el indicador de último registro (LR) antes de finalizar. Si hay una operación de retorno y LR está desactivado, el programa no se desactiva.

A.3 Manejo de Mensajes No Supervisados

Normalmente, si un programa recibe un mensaje de escape no supervisado, el sistema envía el mensaje de comprobación de función (CPF9999) a la cola de mensajes de programa y el programa deja de procesarse. No obstante, los compiladores de programas HLL pueden añadir supervisores para el mensaje de comprobación de función o para los mensajes que puedan aparecer en el programa. (Se enviará un mensaje de consulta a la pantalla de mensajes de programa.) Esto le permite finalizar el programa de la forma que desee. En un trabajo de depuración interactivo, cuando se produce una comprobación de función, el sistema proporciona un manejo por omisión y le da el control en vez de finalizar el programa. El sistema visualiza lo siguiente en la pantalla de mensajes no supervisados:

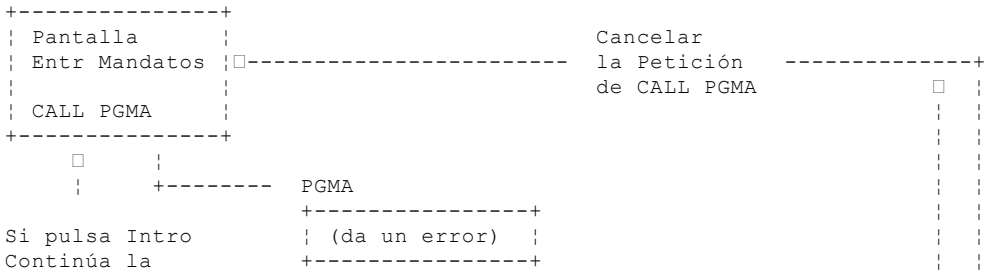
- El mensaje
- El número de instrucción MI y el identificador de sentencia HLL, si están disponible, para los que se envió el mensaje
- El nombre y nivel de recurrencia del programa al que se envió el mensaje

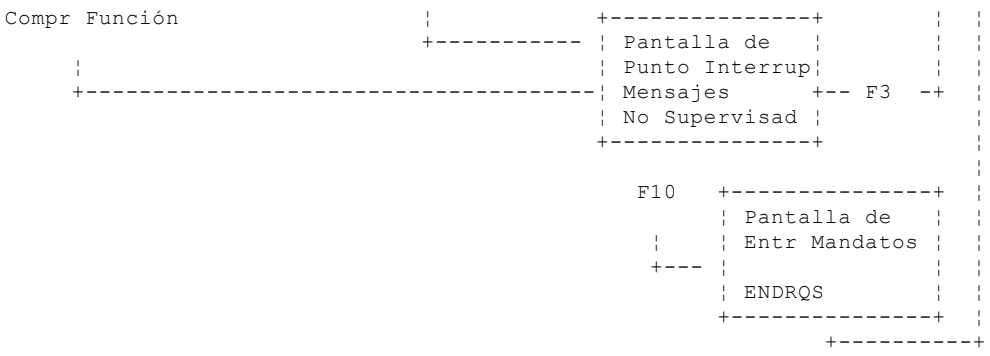
A continuación se facilita un ejemplo de una pantalla de punto de interrupción de mensaje no supervisado:

```
+-------------------------------------------------------------------------------+
|                                                                               |
|                 Visualizar Punto de Interrupción de Mensaje No Supervisado    |
|                                                                               |
|                                                                               |
|   Sentencia/Instrucción . . . . . :   440 /0077                               |
|   Programa  . . . . . . . . . . . :   TETEST                                  |
|   Nivel recurrencia . . . . . . . :   1                                       |
|                                                                               |
|  Errores producidos en el mandato.                                            |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|  Pulse Intro para continuar.                                                  |
|                                                                               |
|  F3=Salir del programa   F10=Entrada de mandatos                              |
|                                                                               |
+-------------------------------------------------------------------------------+
```

Puede intentar identificar el origen del error utilizando las funciones de prueba. Sin embargo, la petición original errónea sigue detenida en el punto en el que se detectó el error. Para eliminar la petición errónea desde la pila de programas, debe utilizar el mandato Finalizar Petición (ENDRQS) o pulsar F3 cuando se visualiza la pantalla de punto de interrupción de mensaje no supervisado. Puede hacer que el proceso normal de comprobación de funciones continúe pulsando la tecla Intro cuando aparece la pantalla de punto de interrupción de mensaje no supervisado. Si pulsa F10 para llamar a la pantalla de entrada de mandatos, debe pulsar F3 para volver a la pantalla de punto de interrupción de mensaje no supervisado.

A continuación se ilustra cómo funciona el mandato ENDRQS:





Las llamadas a programas se destruyen cuando se entra un mandato ENDRQS.
En el diagrama anterior, se destruye la llamada al programa PGMA.)

A.4 Puntos de Interrupción

Un punto de interrupción es una posición en un programa en la el sistema detiene el proceso de un programa y le da el control al usuario de una estación de pantalla (modalidad interactiva) o a un programa especificado en el parámetro BKPPGM en el mandato Añadir Punto de Interrupción (ADDBKP) (modalidad de proceso por lotes).

Subtemas
A.4.1 Adición de Puntos de Interrupción a Programas
A.4.2 Puntos de Interrupción Condicionales
A.4.3 Eliminación de Puntos de Interrupción en los Programas

A.4.1 Adición de Puntos de Interrupción a Programas

Utilice el mandato ADDBKP para añadir puntos de interrupción al programa que desea depurar. Puede especificar 10 identificadores de sentencia como máximo en un solo mandato ADDBKP. Las variables de programa especificadas en un mandato ADDBKP se aplican solamente a los puntos de interrupción especificados en dicho mandato. Puede especificar hasta 10 variables en un mandato ADDBKP.

También puede especificar el nombre del programa al que se va a añadir el punto de interrupción. Si no especifica el nombre del programa al que se desea añadir el punto de interrupción, el punto de interrupción se añade al programa por omisión especificado en el mandato STRDBG, CHGDBG o ADDPGM.

Para obtener más información acerca de mandatos de puntos de interrupción, consulte el manual CL Reference.

Para añadir un punto de interrupción a un programa, especifique un identificador de sentencia, que puede ser:

- Una etiqueta de sentencia
- Un número de sentencia
- Un número de instrucción de la interfaz de máquina (MI)

Cuando añade un punto de interrupción a un programa, también puede especificar las variables de programa cuyos valores o valores parciales desea visualizar cuando se alcance el punto de interrupción. Estas variables pueden mostrarse en formato de caracteres o hexadecimal.

El proceso del programa se detiene en un punto de interrupción *antes* de procesar la instrucción. Para un trabajo interactivo, el sistema visualiza el punto de interrupción en el que el programa se ha detenido y, si se solicita, los valores de las variables del programa.

En los programas de lenguaje de alto nivel, distintas sentencias y etiquetas pueden correlacionarse con la misma instrucción interna. Esto sucede cuando existen varias sentencias inoperantes (tales como DO y ENDDO) una detrás de la otra en un programa. Puede utilizar la lista IRP para determinar qué sentencias o etiquetas se correlacionan con la misma instrucción.

El resultado de que distintas sentencias se correlacionen con la misma instrucción es que un punto de interrupción que se esté añadiendo puede redefinir un punto de interrupción anterior que se había añadido para otra sentencia. Cuando esto sucede, el punto de interrupción nuevo sustituye al añadido anteriormente; es decir, se elimina el punto de interrupción anterior y se añade uno nuevo. Una vez visualizada esta información, puede efectuar una de estas acciones:

- Finalizar la petición más reciente pulsando F3.
- Continuar el proceso del programa pulsando Intro.
- Ir a la pantalla de entrada de mandatos en el siguiente nivel de petición pulsando F10. Desde esta pantalla puede:

  - Entrar cualquier mandato CL que pueda utilizarse en un entorno de depuración interactivo. Puede visualizar o cambiar los valores de las variables en el programa, añadir o eliminar programas de la modalidad de depuración o ejecutar otros mandatos de depuración.

  - Seguir procesando el programa entrando el mandato Reanudar Punto de Interrupción (RSMBKP).

  - Volver a la pantalla de punto de interrupción pulsando F3.

  - Volver a la pantalla de entrada de mandatos en el nivel de petición anterior entrando el mandato Finalizar Petición (ENDRQS).

En el caso de un trabajo por lotes, se puede llamar a un programa de punto de interrupción cuando se llega a un punto de interrupción. Debe crear este programa de punto de interrupción para manejar la información de punto de interrupción. La información de punto de interrupción se pasa al programa de punto de interrupción. El programa de punto de interrupción es otro programa (por ejemplo, un programa CL) que contiene los mismos mandatos (peticiones de funciones) que habría entrado interactivamente para un trabajo interactivo. Por ejemplo, el programa puede visualizar y cambiar variables o añadir y eliminar puntos de interrupción. Se puede solicitar cualquier función que sea válida en un trabajo por lotes. Cuando el programa de punto de interrupción acaba de procesarse, el programa que se estaba depurando continúa.

Se graba un mensaje en las anotaciones de trabajo para cada punto de interrupción del trabajo de depuración.

Los siguientes mandatos ADDBKP añaden punto de interrupción al programa CUS310. CUS310 es el programa por omisión, por lo que no es preciso especificarlo. El valor de la variable &ARBAL se muestra cuando se alcanza el segundo punto de interrupción.

```
ADDBKP  STMT(900)
ADDBKP  STMT(2200)  PGMVAR('&ARBAL')
```

Nota: Se debe entrar una variable CL entre apóstrofos.

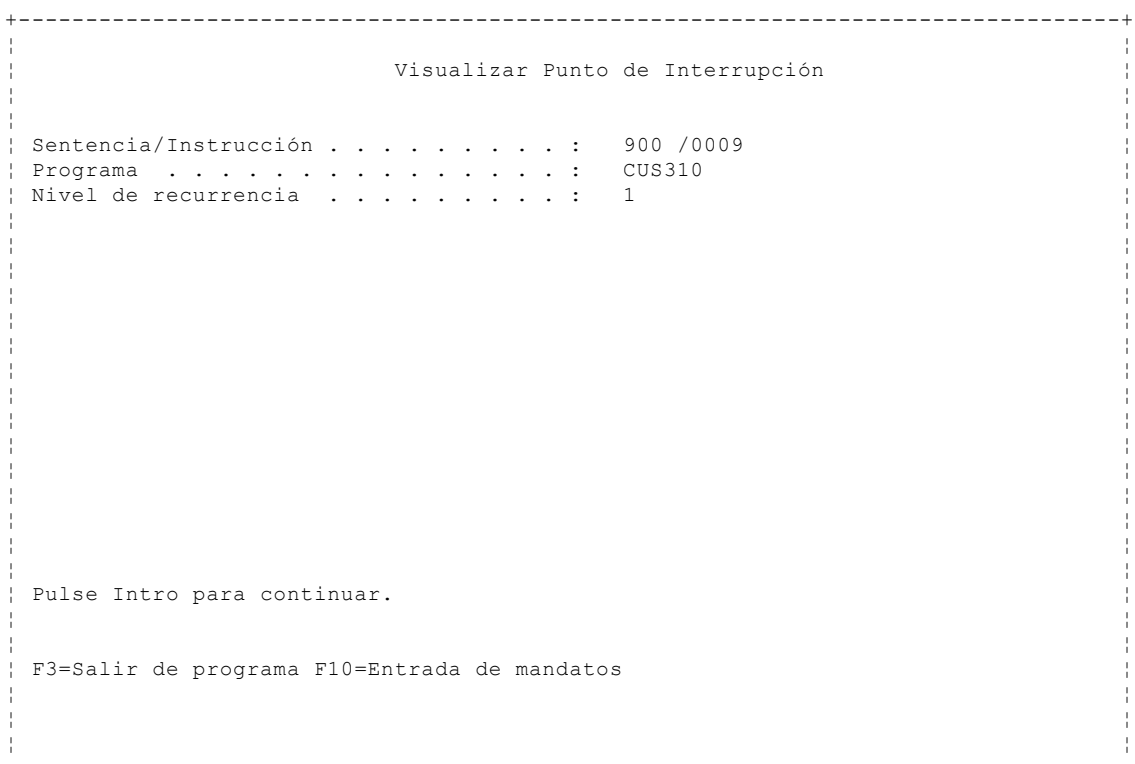
El fuente para CUS310 es el siguiente:

```
5728PW1 R01M00  880101                 LISTADO FUENTE SEU


ARCHIVO FUENTE  . . . . .  QGPL/QCLSRC
MIEMBRO . . . . . .  CUS310

SEQNBR*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...
  100 PGM   PARM(&NBRITEMS &ITEMPRC &PARBAL &PTOTBAL)
  200   DCL VAR(&PARBAL)   TYPE(*DEC) LEN(15 5) /* BALANCE INV AREA ENTRADA  */
  300   DCL VAR(&PTOTBAL)  TYPE(*DEC) LEN(15 5) /* BALANCE INV TOTAL ENTRADA */
  400   DCL VAR(&NBRITEMS) TYPE(*DEC) LEN(15 5) /* NUMERO DE ARTICULOS       */
  500   DCL VAR(&ITEMPRC)  TYPE(*DEC) LEN(15 5) /* PRECIO DEL ARTICULO       */
  600   DCL VAR(&ARBAL)    TYPE(*DEC) LEN(5 2)  /* BALANCE INVENTARIO AREA   */
  700   DCL VAR(&TOTBAL)   TYPE(*DEC) LEN(5 2)  /* BALANCE INVENTARIO TOTAL  */
  800   DCL VAR(&TOTITEM)  TYPE(*DEC) LEN(5 2)  /* PRECIO TOTAL ARTICULOS    */
  900   CHGVAR     VAR(&ARBAL) VALUE(&PARBAL)
 1000   CHGVAR     VAR(&TOTBAL) VALUE(&PTOTBAL)
 1100   IF COND(&NBRITEMS *EQ 0) THEN(DO)
 1200      SNDPGMMSG  MSG('El núm artículos es cero. Efectúe pedido+
 1300                   de este artículo.') TOMSGQ(INVLIB/INVQUEUE)
 1400      GOTO       CMDLBL(EXIT)
 1500   ENDDO
 1600   CHGVAR     VAR(&TOTITEM) VALUE(&NBRITEMS * &ITEMPRC)
 1700   IF COND(&NBRITEMS *GT 50) THEN(DO)
 1800      SNDPGMMSG  MSG('Demasiadas existenc para este artículo.') +
 1900                 TOMSGQ(INVLIB/INVQUEUE)
 2000   ENDDO
 2100   CHGVAR     VAR(&ARBAL)   VALUE(&ARBAL + &TOTITEM)
 2200   IF COND(&ARBAL *GT 1000) THEN(DO)
 2300      SNDPGMMSG  MSG('El área tiene demasiado dinero +
 2400                   en existencias.') TOMSGQ(INVLIB/INVQUEUE)
 2500   ENDDO
 2600   CHGVAR     VAR(&TOTBAL)  VALUE(&TOTBAL + &TOTITEM)
 2700   EXIT:      ENDPGM
```

Como resultado de alcanzar el primer punto de interrupción se visualiza lo siguiente:



```
+-------------------------------------------------------------------------------+
```

Se visualiza lo siguiente como resultado de alcanzar el segundo punto de interrupción:

```
+-------------------------------------------------------------------------------+
|                                                                               |
|                           Visualizar Punto de Interrupción                    |
|                                                                               |
|                                                                               |
| Sentencia/Instrucción . . . . . :   2200 /0022                                |
| Programa  . . . . . :   CUS310                                                |
| Nivel de recurrencia  . . . . . :   1                                         |
| Posición inicial  . . . . . :   1                                             |
| Formato . . . . . :   *CHAR                                                   |
| Longitud  . . . . . :   *DCL                                                  |
|                                                                               |
| Variable  . . . . . :   &ARBAL                                                |
|   Tipo  . . . . . :     PACKED                                                |
|   Longitud  . . . . . :     5 2                                               |
|  '610,00'                                                                     |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
| Pulse Intro para continuar.                                                   |
|                                                                               |
| F3=Salir de programa F10=Entrada de mandatos                                  |
|                                                                               |
|                                                                               |
+-------------------------------------------------------------------------------+
```

Se muestra la variable &ARBAL. (Observe que el valor de &ARBAL variará en función de los valores de los parámetros pasados al programa.) Puede pulsar F10 para visualizar la pantalla de entrada de mandatos para modificar el valor de la variable &ARBAL y alterar así el proceso del programa. Para cambiar el valor de una variable, utilice el mandato Cambiar Variable de Programa (CHGPGMVAR).

A.4.2 Puntos de Interrupción Condicionales

Puede añadir un punto de interrupción condicional a un programa que se está depurando. Utilice el mandato Añadir Punto de Interrupción (ADDBKP) para especificar la sentencia y la condición. Si se satisface la condición, el sistema detiene el proceso del programa en la sentencia especificada.

En el mandato ADDBKP, puede especificar un **valor de salto**, que es un número que indica cuántas veces debe procesarse una sentencia antes de que el sistema detenga el programa. Por ejemplo, para detener un programa en la sentencia 1200 después de que la sentencia se haya procesado 100 veces, entre el mandato siguiente:

```
ADDBKP STMT(1200) SKIP(100)
```

Si especifica varias sentencias al especificar el parámetro SKIP, cada sentencia tiene una cuenta distinta. El mandato siguiente hace que el programa se detenga en la sentencia 150 ó 200, pero solamente después de que la sentencia se haya procesado 400 veces:

```
ADDBKP STMT(150 200) SKIP(400)
```

Si la sentencia 150 se ha procesado 400 veces pero la sentencia 200 sólo se ha procesado 300 veces, el programa no se detiene en la sentencia 200.

Si una sentencia no se ha procesado todas las veces que se ha especificado en el parámetro SKIP, puede utilizarse el mandato Visualizar Punto de Interrupción (DSPBKP) para mostrar cuántas veces se ha procesado la sentencia. Para restaurar a cero la cuenta de SKIP para una sentencia, entre de nuevo el punto de interrupción para dicha sentencia.

Puede especificar una condición de punto de interrupción más general en el mandato ADDBKP. Esta expresión utiliza una variable de programa, un operador y otra variable o constante como operandos. Por ejemplo, para detener un programa en la sentencia 1500 cuando la variable &X es superior a 1000, entre el mandato siguiente:

```
ADDBKP STMT(1500) PGMVAR('&X') BKPCOND(*PGMVAR1 *GT 1000)
```

El parámetro BKPCOND requiere tres valores:

- En el ejemplo, el primer valor especifica la primera variable especificada en el parámetro PGMVAR. (Para especificar la tercera variable, se utilizaría *PGMVAR3.)

- El segundo valor debe ser un operador. Para ver una lista de todos los operadores válidos, consulte el manual CL Reference.

- El tercer valor puede ser una constante u otra variable. Una constante puede ser un número, una serie de caracteres, una serie de bits y debe ser del mismo tipo que la variable de programa especificada en el primer valor.

Los parámetros SKIP y BKPCOND pueden utilizarse juntos para especificar una condición de punto de interrupción compleja. Por ejemplo, para detener un programa en la sentencia 1000 después de que esta sentencia se haya procesado 50 veces **y** únicamente cuando la serie de caracteres &STR sea **TRUE**, introduzca el mandato siguiente:

```
ADDBKP STMT(1000) PGMVAR('&STR') SKIP(50)
       BKPCOND(*PGMVAR1 *EQ 'TRUE ')
```

A.4.3 Eliminación de Puntos de Interrupción en los Programas

Para eliminar puntos de interrupción de un programa, utilice el mandato Eliminar Punto de Interrupción (RMVBKP). Para ello, debe especificar el número de la sentencia para al que se ha definido el punto de interrupción.

A.5 Rastreos

Un rastreo es el proceso de registrar la secuencia en la que se procesan las sentencias de un programa. Se diferencia de un punto de interrupción en que el usuario no recibe el control durante el rastreo. El sistema registra las sentencias rastreadas que se han procesado. Sin embargo, la información de rastreo no se visualiza automáticamente cuando el programa completa el proceso. Hay que solicitar la pantalla de información de rastreo utilizando el mandato Visualizar Datos de Rastreo (DSPTRCDTA). La pantalla muestra la secuencia en la que se han procesado las sentencias y, si se solicita, los valores de las variables especificadas en el mandato Añadir Rastreo (ADDTRC).

Subtemas

A.5.1 Adición de Rastreos a Programas

La adición de un rastreo consiste en especificar qué sentencias van a rastrearse y, si se desea, los nombres de las variables de programa. Antes de procesar una sentencia rastreada, se registra el valor de la variable. Asimismo, puede especificar que los valores de las variables se registren solamente si han cambiado desde la última vez que se procesó una sentencia rastreada. Estas variables pueden visualizarse en formato de caracteres o en formato hexadecimal.

Para especificar qué sentencias van a rastrearse, puede especificar:

- El identificador de sentencia en el que empezará el rastreo y el identificador de sentencia en el que se detendrá
- Que se rastreen todas las sentencias de un programa
- Un identificador de sentencia único de una sentencia que vaya a rastrearse

En los mandatos STRDBG o CHGDBG, puede especificar cuántos rastreos de sentencias se pueden registrar para un trabajo y qué acción debe emprender el sistema cuando se llega al número máximo. Cuando se alcanza el número máximo, el sistema efectúa una de estas acciones (en función de lo que haya especificado):

- Para un trabajo interactivo, puede realizarse una de las siguientes acciones:
  - Detener el rastreo (*STOPTRC). Se le transfiere el control (se produce un punto de interrupción), y puede eliminar algunas de las definiciones de rastreo (mandato RMVTRC), borrar los datos de rastreo (mandato CLRTRCDTA) o cambiar el número máximo (parámetro MAXTRC en el mandato CHGDBG).
  - Continuar el rastreo (*WRAP). Los datos de rastreo registrados anteriormente se recubren con los datos de rastreo registrados después de este punto.
- Para un trabajo por lotes, puede realizarse una de estas acciones:
  - Detener el rastreo (*STOPTRC). Las definiciones de rastreo se eliminan y el programa sigue procesándose.
  - Continuar el rastreo (*WRAP). Los datos de rastreo registrados anteriormente se recubren con los datos de rastreo registrados después de este punto.

Puede cambiar el máximo y la acción por omisión en cualquier momento durante el trabajo de depuración utilizando el mandato Cambiar Depuración (CHGDBG). Sin embargo, el cambio no afecta a los rastreos que ya han sido registrados.

Puede especificar un total de cinco rangos de sentencias para un solo programa en cualquier momento. Este total se toma de todos los mandatos Añadir Rastreo (ADDTRC) del programa. Además, sólo pueden especificarse 10 variables para cada rango de sentencias.

En los programas de lenguaje de alto nivel, distintas sentencias y etiquetas pueden correlacionarse con la misma instrucción interna. Esto sucede cuando existen varias sentencias inoperantes (tales como DO y END) una detrás de la otra en un programa. Puede utilizar la lista IRP para determinar qué sentencias o etiquetas se correlacionan con la misma instrucción.

Cuando especifique variables CL, debe colocar el símbolo & y el nombre de la variable entre apóstrofos. Por ejemplo:

```
  ADDTRC PGMVAR('&IN01')
```

Cuando se especifica un rango de sentencia, el número de sentencia fuente para la sentencia de parada suele ser mayor que el número de la sentencia inicial. El rastreo, sin embargo, se realiza con las instrucciones de la interfaz de máquina (MI), y algunos compiladores (especialmente RPG/400) generan programas en los que el orden de las instrucciones MI no es el mismo que el de las sentencias fuente. Por consiguiente, en algunos casos, el número MI de la sentencia de parada no puede ser superior al número MI de la sentencia inicial, y recibirá el mensaje CPF1982.

Cuando reciba este mensaje, debe realizar una de las acciones siguientes:

- Rastrear todas la sentencias del programa.
- Restringir un rango de sentencias a una especificación.
- Utilizar los números de instrucción MI obtenidos de una representación intermedia de un listado del programa (IRP). (Véase el apartado "Depuración a Nivel de Interfaz de Máquina" en el tema A.10.)

El siguiente mandato Añadir Rastreo (ADDTRC) añade un rastreo al programa CUS310. CUS310 es el programa por omisión, por lo que no es preciso especificarlo. El valor de la variable &TOTBAL se registra solamente si su valor cambia desde que se procesa cada sentencia rastreada hasta que se vuelve a procesar.

```
ADDTRC STMT((900 2700))  PGMVAR('&TOTBAL')  OUTVAR(*CHG)
```

Las siguientes pantallas son el resultado de este rastreo, y se visualizan utilizando el mandato Visualizar Datos de Rastreo (DSPTRCDTA). Observe que no se proporcionan cabeceras de columna para todas las pantallas.

```
                          Visualizar Datos de Rastreo

                 Sentencia/
Programa         Instrucción           Nivel recurrencia     Núm secuencia
CUS310           900                        1                          1

 Posición inicial  . . . . . . . . . . . :   1
 Longitud  . . . . . . . . . . . . . . . :   *DCL
 Formato . . . . . . . . . . . . . . . . :   *CHAR

 Variable  . . . . . . . . . . . . . . . :   &TOTBAL
   Tipo  . . . . . . . . . . . . . . . . :     PACKED
   Longitud  . . . . . . . . . . . . . . :     5 2
  '    ,00'

                 Sentencia/
Programa         Instrucción           Nivel recurrencia     Núm secuencia
CUS310           1000                       1                          2
CUS310           1100                       1                          3  +

Pulse Intro para continuar.

F3=Salir  F12=Cancelar
```

```
                          Visualizar Datos de Rastreo


 Posición inicial  . . . . . . . . . . . :   1
 Longitud  . . . . . . . . . . . . . . . :   *DCL
 Formato . . . . . . . . . . . . . . . . :   *CHAR

*Variable  . . . . . . . . . . . . . . . :   &TOTBAL
   Tipo  . . . . . . . . . . . . . . . . :     PACKED
   Longitud  . . . . . . . . . . . . . . :     5 2
  '   1,00'

                 Sentencia/
Programa         Instrucción           Nivel recurrencia     Núm secuencia
CUS310           1600                       1                          4
CUS310           1700                       1                          5
CUS310           2100                       1                          6
CUS310           2200                       1                          7
CUS310           2600                       1                          8  +

Pulse Intro para continuar.

F3=Salir  F12=Cancelar
```

```
                          Visualizar Datos de Rastreo

CUS310           2700                       1                          9
```

```
 Posición inicial  . . . . . :   1
 Longitud  . . . . . :   *DCL
 Formato . . . . . :   *CHAR

*Variable  . . . . . :   &TOTBAL
   Tipo  . . . . . :     PACKED
   Longitud  . . . . . :     5 2
  '   2,00'




Pulse Intro para continuar.

F3=Salir  F12=Cancelar
```

A.5.2 Instrucciones Paso a Paso

Puede pasar por cada una de las instrucciones de un programa utilizando los mandatos STRDBG o CHGDBG y estableciendo el parámetro MAXTRC en 1 y el parámetro TRCFULL en *STOPTRC. Cuando especifica un rango de rastreo (mandato ADDTRC) y el programa procesa una instrucción dentro de ese rango, aparece una pantalla de punto de interrupción con un mensaje de error. Si pulsa Intro, se visualiza otra pantalla de punto de interrupción con el mismo mensaje de error para la siguiente instrucción procesada en el rango de rastreo. Cuando finaliza el rastreo, los datos del mismo contienen una lista de las instrucciones rastreadas. Puede visualizar estos datos entrando el mandato Visualizar Datos de Rastreo (DSPTRCDTA).

A.5.3 Utilización de Puntos de Interrupción en Rastreos

Los punto de interrupción pueden utilizarse en un rango de rastreo. En un punto de interrupción dentro de un rastreo, puede visualizar los datos de rastreo (mandato DSPTRCDTA) para determinar si es necesario que lleve a cabo alguna acción. Los datos de rastreo se registran antes del punto de interrupción. La información de rastreo contiene los valores de las variables *antes* de que se procesara la sentencia.

A.5.4 Eliminación de Información de Rastreo del Sistema

En el mandato DSPTRCDTA, puede especificar si la información de rastreo se eliminará del sistema o si permanecerá en el sistema una vez que se haya visualizado. Si deja la información de rastreo en el sistema, se le añadirán otros rastreos. La información permanece en el sistema (a menos que se elimine) hasta que finaliza el trabajo de depuración o hasta que se somete el mandato ENDDBG. También puede utilizar el mandato Borrar Datos de Rastreo (CLRTRCDTA) para eliminar la información de rastreo del sistema.

A.5.5 Eliminación de Rastreos de Programas

El mandato Eliminar Rastreo (RMVTRC) elimina todos o alguno de los rangos especificados en uno o varios mandatos Añadir Rastreo (ADDTRC). Eliminar un rango de rastreo consiste en especificar los identificadores de sentencia utilizados en el mandato RMVTRC o en especificar la eliminación de todos los rangos.

Puede utilizar el parámetro STMT en el mandato RMVTRC para especificar:

- Que no se rastreen todas las sentencias HLL y/o las instrucciones de máquina del programa especificado independientemente de cómo el mandato ADDTRC definió el rastreo.

- La ubicación de arranque y parada de rastreo de las sentencias HLL y/o las instrucciones del sistema a eliminar.

Los mandatos RMVPGM y ENDDBG también eliminan rastreos, pero hacen que el programa deje de estar en modalidad de depuración.

A.6 Funciones de Visualización

En la modalidad de depuración, puede visualizar información de prueba que le permita revisar cómo ha establecido el trabajo de depuración. Puede visualizar qué programas están en modalidad de depuración y qué puntos de interrupción y qué rastreos se han definido para estos programas. Además, puede visualizar el estado de los programas que están en modalidad de depuración.

Para visualizar la información de prueba, puede utilizar los mandatos siguientes:

- Visualizar Depuración (DSPDBG), que muestra la pila de llamadas actual y los nombres de los programas que están en modalidad de depuración e indica lo siguiente:

  - Cuáles están detenidos en un punto de interrupción
  - A cuáles se llaman actualmente
  - El nivel de petición de los que han sido llamados
  - Las opciones de depuración seleccionadas para el trabajo de depuración

- Visualizar Puntos de Interrupción (DSPBKP), que visualiza la ubicación de los puntos de interrupción definidos actualmente en un programa.

- Visualizar Rastreo (DSPTRC), que visualiza las sentencias o los rangos de sentencias definidos actualmente en un programa.

A.7 Visualización de los Valores de las Variables

Cuando se encuentra en un punto de interrupción, puede visualizar los valores de variables de programa. Puede hacerlo automáticamente en la pantalla de punto de interrupción especificando los nombres de variable en el mandato ADDBKP, o bien puede entrar el mandato Visualizar Variable de Programa (DSPPGMVAR) en el punto de interrupción pulsando F10 para mostrar la pantalla de entrada de mandatos. Solo pueden especificarse 10 variables en un mandato DSPPGMVAR. Para variables de carácter y de bit, puede indicar al sistema que empiece a visualizar el valor de la variable a partir de una determinada posición y para una longitud determinada. Las variables pueden visualizarse en formato carácter o hexadecimal.

Notas:

1. Si especifica una variable matriz, puede realizar una de las siguientes acciones:

   a. Especificar los valores de subíndice del elemento matriz que desea visualizar. Los valores de subíndice pueden ser valores enteros o los nombres de variables numéricas del programa.
   b. Visualizar la matriz completa no introduciendo subíndice alguno.
   c. Visualizar una sección cruzada de una sola dimensión de la matriz especificando los valores para todos los subíndices excepto uno y un asterisco para este valor de subíndice.

2. Los nombres de las variables pueden especificarse como simples o calificados, pero deben colocarse entre apóstrofos. Un nombre calificado puede especificarse de una de estas dos formas:

   a. Nombres de variables que se alternan con las palabras especiales separadoras OF o IN, clasificadas de menor a mayor nivel de calificación. Un espacio en blanco debe separar el nombre de la variable y la palabra especial separadora.
   b. Nombres de variables separadas por puntos, clasificadas de mayor a menor nivel de calificación.

El siguiente mandato DSPPGMVAR visualiza la variable ARBAL utilizada en el programa CUS310. CUS310 es el programa por omisión, por lo que no es preciso especificarlo. El valor entero se visualizara en formato de caracteres.

```
  DSPPGMVAR  PGMVAR('&ARBAL')
```

La pantalla resultante es la siguiente:

```
                        Visualizar Variables de Programa

 Programa  . . . . . . . . . . . . . . . :   CUS310
 Nivel de recurrencia  . . . . . . . . . :   1
 Posición inicial  . . . . . . . . . . . :   1
 Formato . . . . . . . . . . . . . . . . :   *CHAR
 Longitud  . . . . . . . . . . . . . . . :   *DCL

 Variable  . . . . . . . . . . . . . . . :   &ARBAL
   Tipo  . . . . . . . . . . . . . . . . :   PACKED
   Longitud  . . . . . . . . . . . . . . :   5 2
  '610,00'









 Pulse Intro para continuar.


 F3=Salir  F12=Cancelar
```

Algunos programas HLL permiten que las variables se basen en una variable de puntero especificada por el usuario (puntero HLL). Si no especifica un puntero explícito para una variable que se basa en otra, se utiliza el puntero especificado en la declaración HLL (si lo hay). Debe especificar un puntero de base explícito si no hay ninguno especificado en la declaración para la variable que se basa en otra. El parámetro PGMVAR le

permite especificar hasta cinco punteros de base explícitos al hacer referencia a una variable basada. Cuando se especifican varios punteros de base, el primero de ellos se utiliza para localizar el segundo, el segundo para localizar el tercero, y así sucesivamente. El último puntero de la lista se utiliza para localizar la variable primaria.

A.8 Cambio de los Valores de las Variables

Para cambiar el valor de una variable de programa, utilice el mandato Cambiar Variable de Programa (CHGPGMVAR), el mandato Cambiar Puntero HLL (CHGHLLPTR) o Cambiar Puntero (CHGPTR). Cambiar el valor de una variable de programa consiste en especificar el nombre de variable y un valor que sea compatible con el tipo de datos de la variable. Por ejemplo, si la variable es de tipo carácter, debe entrar un valor de tipo carácter.

Al cambiar el valor de las variables, hay que tener en cuenta si la variable es automática o estática. La diferencia entre ambas radica en el almacenamiento de las variables. Para las variables automáticas, el almacenamiento se asocia con la llamada del programa. Cada vez que se llama a un programa, una nueva copia de la variable se sitúa en el almacenamiento automático. Un cambio efectuado a una variable automática permanece en vigor solamente para la llamada del programa en el que se realizó dicho cambio.

Nota: En algunos lenguajes, la definición de una llamada se realiza a nivel de procedimiento y no solamente a nivel de programa. Para estos lenguajes, el almacenamiento de variables automáticas está asociado con la llamada al procedimiento. Cada vez que se llama a un procedimiento, se obtiene una nueva copia de la variable. Un cambio efectuado en una variable automática permanece en vigor solamente mientras se llama a dicho procedimiento. Solamente pueden cambiarse las variables automáticas en las llamadas de procedimiento más recientes. El parámetro RCRLVL (nivel de recurrencia) de los mandatos se aplica solamente en base a un programa y no en base a un procedimiento.

Para variables estáticas, el almacenamiento se asocia con la activación. Solo existe una copia de una variable estática en el almacenamiento independientemente de cuántas veces se llama a un programa. Un cambio en una variable estática permanece en vigor mientras dura la activación.

Para determinar si una variable de programa es estática o automática, solicite una lista de IRP (representación intermedia de un programa) (*LIST y *XREF en el parámetro GENOPT) cuando se cree el programa que contiene las variables.

Al cambiar una variable que es una matriz, hay que especificar un elemento de la matriz. Por consiguiente, hay que especificar los valores de subíndice para el elemento matriz que se desee cambiar.

A.9 Utilización de un Trabajo para Depurar Otro Trabajo

Es posible que desee utilizar un trabajo por separado para depurar programas que se ejecutan en otro trabajo por uno de estos motivos:

- Un trabajo interactivo puede depurar trabajos por lotes.
- Un trabajo interactivo puede depurarse desde otro trabajo interactivo. Esto permite que una pantalla muestre la información de depuración sin interrumpir la pantalla del programa de aplicación.
- Un trabajo interactivo o por lotes que se repita en bucle puede interrumpirse y colocarse en modalidad de depuración.

Subtemas

A.9.1 Depuración de Trabajos por Lotes Sometidos a una Cola de Trabajos

La utilización de un trabajo por separado para depurar otro trabajo por lotes sometido a la cola de trabajos le permite colocar el trabajo por lotes en modalidad de depuración y establecer puntos de interrupción y rastreos antes de que el trabajo empiece a procesarse. Utilice los pasos siguientes para depurar trabajos por lotes que se van a someter a una cola de trabajos:

1. Someta el trabajo por lotes utilizando el mandato Someter Trabajo (SBMJOB) o un programa que somete automáticamente el trabajo con HOLD(*YES).

   ```
   SBMJOB HOLD(*YES)
   ```

2. Determine el nombre calificado (número/usuario/nombre) que está asignado al trabajo mediante el mandato Trabajar con Trabajos Sometidos (WRKSBMJOB) o el mandato Trabajar con Colas de Trabajos (WRKJOBQ). El mandato SBMJOB también visualiza el nombre en un mensaje de terminación cuando el mandato finaliza el proceso.

   El mandato WRKJOBQ (Trabajar con Cola de Trabajos) visualiza todos los trabajos que esperan el arranque en una cola de trabajos determinada. Puede mostrar el nombre del trabajo desde esta pantalla seleccionando la opción 5 para el trabajo.

3. Entre el mandato Arrancar Trabajo de Servicio (STRSRVJOB) desde la pantalla que piensa utilizar para depurar el trabajo por lotes de la forma siguiente:

   ```
   STRSRVJOB JOB (nombre-calificado-trabajo)
   ```

4. Entre el mandato STRDBG y proporcione los nombres de todos los programas que van a depurarse. No puede entrar ningún otro mandato de depuración mientras el trabajo espera en la cola de trabajos.

5. Utilice el mandato Liberar Cola de Trabajos (RLSJOBQ) para liberar la cola de trabajos. Aparece una pantalla cuando el trabajo está preparado para arrancar, indicando que puede empezar a depurar el trabajo. Pulse F10 para mostrar la pantalla Entrada de Mandatos.

6. Utilice la pantalla Entrada de Mandatos para entrar los mandatos de depuración, tales como Añadir Punto de Interrupción (ADDBKP) o Añadir Rastreo (ADDTRC).

7. Pulse F3 para salir de la pantalla Entrada de Mandatos y luego pulse Intro para arrancar el trabajo por lotes.

8. Cuando el trabajo se detiene en un punto de interrupción, se visualiza la pantalla normal de punto de interrupción. Cuando finaliza el trabajo, no puede añadir puntos de interrupción y rastreos ni visualizar o cambiar variables. Sin embargo, puede visualizar los datos de rastreo utilizando el mandato Visualizar Datos de Rastreo (DSPTRCDTA).

9. Si desea depurar otro trabajo por lotes, finalice primero la depuración utilizando el mandato Finalizar Depuración (ENDDBG) y finalice luego el servicio del trabajo utilizando el mandato Finalizar Trabajo de Servicio (ENDSRVJOB).

A.9.2 Depuración de Trabajos por Lotes No Arrancados desde Colas de Trabajos

Algunos trabajos arrancados en el sistema no se someten a una cola de trabajos. Estos trabajos no se pueden detener antes de que comiencen a ejecutarse, pero generalmente se pueden depurar. Para depurar trabajos que no se han arrancado desde una cola de trabajos, efectúe lo siguiente:

1. Cambie el nombre del programa al que se llama cuando se arranca el trabajo. Por ejemplo, si el trabajo ejecuta el programa CUST310, puede cambiar el nombre de este programa por CUST310DBG.

2. Cree un pequeño programa CL con el mismo nombre que el programa original (antes de cambiarle el nombre). En este pequeño programa CL, utilice el mandato Retardar Trabajo (DLYJOB) para especificar un retardo de un minuto y después utilice el mandato CALL para llamar al programa renombrado.

3. Permita que el trabajo por lotes se arranque para retardar un minuto el programa CL.

4. Utilice el mandato Trabajar con Trabajos Activos (WRKACTJOB) para buscar el trabajo por lotes que está ejecutándose. Cuando aparezca la pantalla, introduzca la opción 5 junto al trabajo para obtener el nombre de trabajo calificado.

5. Entre el mandato Arrancar Trabajo de Servicio (STRSRVJOB) de la forma siguiente:

   ```
   STRSRVJOB JOB (nombre-calificado-trabajo)
   ```

6. Introduzca STRDBG y cualquier otro mandato de depuración, como Añadir Punto de Interrupción (ADDBKP) o Añadir Rastreo (ADDTRC). Continúe con la depuración como es habitual.

A.9.3 Depuración de un Trabajo en Ejecución

Puede depurar un trabajo que se está ejecutando si sabe las sentencias que va a ejecutar el trabajo. Por ejemplo, puede que quiera depurar un programa en ejecución si el trabajo está en un bucle o si el trabajo todavía no ejecuta un programa que ha de depurarse. Los siguientes pasos le permiten depurar un trabajo en ejecución:

1. Utilice el mandato Trabajar con Trabajos Activos (WRKACTJOB) para buscar el trabajo que está ejecutándose. Cuando aparezca la pantalla, introduzca la opción 5 junto al trabajo para obtener el nombre de trabajo calificado.

2. Entre el mandato Arrancar Trabajo de Servicio (STRSRVJOB) de la forma siguiente:

   ```
   STRSRVJOB JOB (nombre-calificado-trabajo)
   ```

3. Entre el mandato Iniciar Depuración (STRDBG) (con ello no se detendrá la ejecución del trabajo).

   Nota: Puede utilizar el mandato Visualizar Depuración (DSPDBG) para visualizar la pila de llamadas. Sin embargo, a menos que el programa se detenga por alguna razón, la pila de llamadas es correcta sólo durante un instante, y el programa sigue ejecutándose.

4. Si sabe una de las sentencias que va a ejecutarse, entre el mandato Añadir Punto de Interrupción (ADDBKP) para detener el trabajo en dicha sentencia.

   Si no sabe qué sentencias están ejecutándose, haga lo siguiente:

   a. Entre el mandato Añadir Rastreo (ADDTRC).

   b. Tras un breve espacio de tiempo, entre el mandato Eliminar Rastreo (RMVTRC) para detener el rastreo del programa.

   c. Entre el mandato Visualizar Datos de Rastreo (DSPTRCDTA) para mostrar las sentencias que se han procesado. Utilice los datos de rastreo para determinar qué sentencias de datos se procesan a continuación (por ejemplo, las sentencias dentro de un bucle de programa).

   d. Entre el mandato Añadir Punto de Interrupción (ADDBKP) para detener el trabajo en la sentencia.

5. Entre los mandatos de depuración que desee cuando el programa se detenga en un punto de interrupción.

A.9.4 Depuración de Otro Trabajo Interactivo

Puede depurar un trabajo desde otra pantalla, tanto si el trabajo se está ejecutando o si está esperando en una pantalla de entrada de mandato o menú. Para depurar otro trabajo interactivo, haga lo siguiente:

1. Determine el nombre calificado del trabajo que va a depurarse. Para determinar el nombre, entre el mandato Visualizar Trabajo (DSPJOB) desde la pantalla del trabajo que va a depurarse o utilice el mandato Trabajar con Trabajos Activos (WRKACTJOB).

2. Entre el mandato Arrancar Trabajo de Servicio (STRSRVJOB) utilizando el nombre del trabajo calificado.

3. Entre el mandato Iniciar Depuración (STRDBG) y cualquier otro mandato de depuración que desee. Si el trabajo ya está ejecutándose, tal vez tenga que entrar el mandato Visualizar Depuración (DSPDBG) para determinar qué sentencia del programa está procesándose.

Cuando el trabajo que está depurándose se detiene en un punto de interrupción, la estación de pantalla se bloquea.

A.9.5 Consideraciones al Depurar Un Trabajo desde Otro Trabajo

Aunque la mayoría de trabajos pueden se pueden depurar desde otro trabajo, debe tener en cuenta las siguientes consideraciones:

- Un trabajo en depuración no puede retenerse o cancelarse (por ejemplo, al ejecutar otro trabajo de grupo o un trabajo secundario).

- Cuando se da servicio a otro trabajo con el mandato Arrancar Trabajo de Servicio (STRSRVJOB), tampoco se puede depurar el trabajo que está dando servicio. Todos los mandatos de depuración se aplican solamente al trabajo al que se está dando servicio. Para depurar el trabajo que da servicio, hay que finalizar el servicio del otro trabajo o hacer que otro trabajo le dé servicio y depurarlo.

- Los mandatos de depuración operan en otro trabajo, incluso si éste trabajo no se detiene en un punto de interrupción. Por ejemplo, si está depurándose un trabajo en ejecución y se entra el mandato Visualizar Variable de Programa (DSPPGMVAR), se muestra la variable que se especifique. Puesto que el trabajo continúa la ejecución, el valor de la variable puede cambiar poco después de que se introduzca el mandato.

- Un trabajo en depuración debe tener prioridad suficiente para responder a los mandatos de depuración. Si está depurándose un trabajo por lotes con prioridad baja y este trabajo no obtiene ningún tiempo de proceso, entonces cualquier mandato de depuración que se emita espera una respuesta del trabajo. Si el trabajo no responde, el mandato finaliza y se visualiza un mensaje de error.

- No puede dar servicio y depurar un trabajo que está depurándose a sí mismo. Sin embargo, puede dar servicio y depurar a un trabajo que está dando servicio y depurando a otro trabajo.

A.10 Depuración a Nivel de Interfaz de Máquina

Para depurar sus programas a nivel de interfaz de máquina (MI), puede especificar un número de vector de definición de objeto MI (ODV) en el parámetro PGMVAR de un mandato y números de instrucción MI en el parámetro STMT de un mandato. Para un punto de interrupción, el sistema se detiene en el número de instrucción MI de la misma forma que lo haría en un número de sentencia HLL. El ODV o la instrucción MI siempre han de ir precedidos de una barra inclinada (/) y deben ir entre apóstrofos (por ejemplo, **'/1A'**) para indicar al sistema que está depurando al nivel MI.

Los números de instrucción MI y ODV pueden obtenerse del listado IRP producido por la mayoría de los compiladores de lenguaje de alto nivel. Utilice el valor *LIST del parámetro GENOPT para producir el listado IRP en el momento al crear el programa.

Nota: Cuando depura a nivel de interfaz de máquina, solamente están disponibles las características que están definidas a nivel de interfaz de máquina; las características HLL que normalmente se transfieren al entorno de prueba no están disponibles. Estas características HLL pueden incluir: tipo de variable, número de dígitos fraccionarios, longitud e información de matriz. Por ejemplo, una variable numérica en el programa HLL puede visualizarse sin la correcta alineación decimal o posiblemente como una serie de caracteres.

A.11 Consideraciones de Seguridad

Para depurar un programa, debe tener autorización *CHANGE para este programa. La autorización *CHANGE disponible mediante la adopción de otro perfil de usuario no se tiene en cuenta al determinar si un usuario tiene autorización para depurar un programa. Esto impide a los usuarios acceder a los datos de un programa en modalidad de depuración adoptando otro perfil de usuario.

De forma adicional, cuando esté en un punto de interrupción definido por el usuario de un programa que está depurando con una autorización de usuario adoptada, sólo tiene la autorización de su perfil de usuario y no la del perfil adoptado. No dispone de las autorizaciones adoptadas por las llamadas anteriores del programa para todos los puntos de interrupción si éstos han sido añadidos con el mandato Añadir Punto de Interrupción (ADDBKP) o han sido causados por un mensaje de escape no supervisado.

Subtemas
A.11.1 Utilización de COPY, SAVE, RESTORE, CRTDUPOBJ y CHKOBJITG durante la Depuración

A.11.1 Utilización de COPY, SAVE, RESTORE, CRTDUPOBJ y CHKOBJITG durante la Depuración

Los puntos de interrupción o los rastreos de sentencias pueden **suprimirse** **temporalmente** de un programa mientras se ejecuta la función de depuración, si el usuario utiliza determinados mandatos de lenguaje de control (CL) para especificar su biblioteca o programa. Los puntos de interrupción y los rastreos de sentencias se *restauran* cuando el mandato Cl finaliza su ejecución. Aparecerá un mensaje CPD190A en las anotaciones de trabajo cuando se supriman los puntos de interrupción o los rastreos de sentencias, y otro cuando se restauren.

Los puntos de interrupción o los rastreos de sentencias pueden **suprimirse** **temporalmente** del programa cuando se utilicen los siguientes mandatos CL para especificar su biblioteca:

CHKOBJITG	CPY	CPROBJ	RSTLIB	SAVLIB
	CPYLIB	CRTDUPOBJ	RSTOBJ	SAVOBJ
				SAVSYS
				SAVCHGOBJ

Nota: Cuando los mandatos CL se ejecutan en su programa, no debe añadir puntos de interrupción ni rastreos. Si introduce el mandato Añadir Punto de Interrupción (ADDBKP) o el mandato Añadir Rastreo (ADDTRC), cuando algun mandato se ejecuta en el programa, recibirá el mensaje de error CPF7102.

B.0 Apéndice B. Mandato TFRCTL

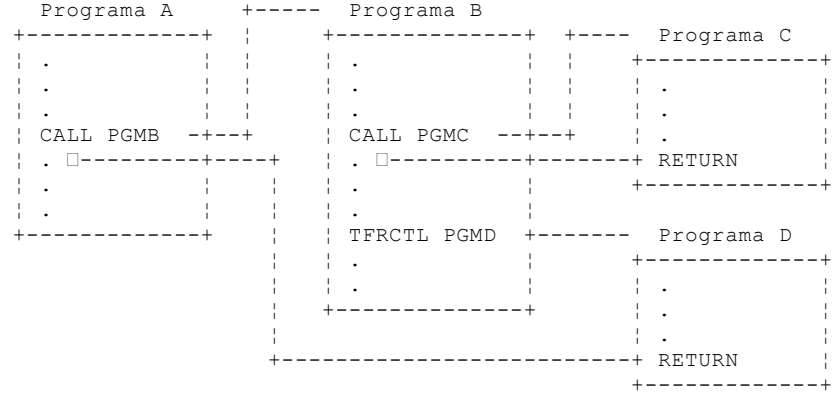
El mandato Transferir Control (TFRCTL) llama al programa especificado en el mandato, le pasa el control y suprime el programa que ha transferido el control de la pila de llamadas. La reducción del número de programas en la pila de llamadas puede mejorar el rendimiento. Cuando se utiliza un mandato CALL, el programa al que se llama devuelve el control al programa que contiene dicho mandato. Cuando se utiliza un mandato TFRCTL, el control vuelve al primer programa de la pila de llamadas. Entonces el primer programa inicia la siguiente instrucción de secuencia que sigue al mandato CALL.

Nota: El mandato TRFCTL no es válido en procedimientos ILE CL.

Subtemas
B.1 Utilización del Mandato TFRCTL
B.2 Paso de Parámetros

B.1 Utilización del Mandato TFRCTL

En la siguiente ilustración, si se especifica un Programa A con USRPRF(*OWNER), la autorización del propietario está en vigor para todos los programas mostrados. Si se especifica un programa B con USRPRF(*OWNER), la autorización del propietario sólo está en vigor mientras los programas B y C están activos. Cuando el Programa B transfiere el control al Programa D, B ya no está en la pila de llamadas y su propietario ya no tiene autorización durante la ejecución del Programa D. Cuando los programas finalizan el proceso (mediante la devolución o la transferencia del control), las autorizaciones del propietario dejan de tener efecto. Todas las alteraciones realizadas por el programa B permanecen en vigor mientras se ejecuta el programa D y se pierden en el momento en que se ejecuta el retorno en el programa D.



El mandato TFRCTL tiene el formato siguiente:

```
TFRCTL PGM(nombre-biblioteca/nombre-programa) PARM(variable-CL)
```

El programa (y calificador de biblioteca) puede ser una variable.

Es importante observar que sólo pueden utilizarse variables como argumentos de parámetros en este mandato y que dichas variables deben haberse recibido como un parámetro en la lista de argumentos del programa que llamó al programa que transfiere el control. Es decir, el mandato TFRCTL no puede pasar una variable que no se haya pasado al programa que ejecuta el mandato TFRCTL.

En el ejemplo siguiente, el primer TRFCTL es válido. El segundo no lo es debido a que &B no se pasó a este programa. El tercer mandato TFRCTL no es válido porque una constante no puede especificarse como argumento.

```
PGM PARM(&A)
DCL &A *DEC 3
DCL &B *CHAR 10
IF (&A *GT 100) THEN (TFRCTL PGM(PGMA) PARM(&A)) /*   válido  */
IF (&A *GT 50) THEN (TFRCTL PGM(PGMB) PARM(&B))  /* no válido */
ELSE (TFRCTL PGM(PGMC) PARM('1'))                /* no válido */
ENDPGM
```

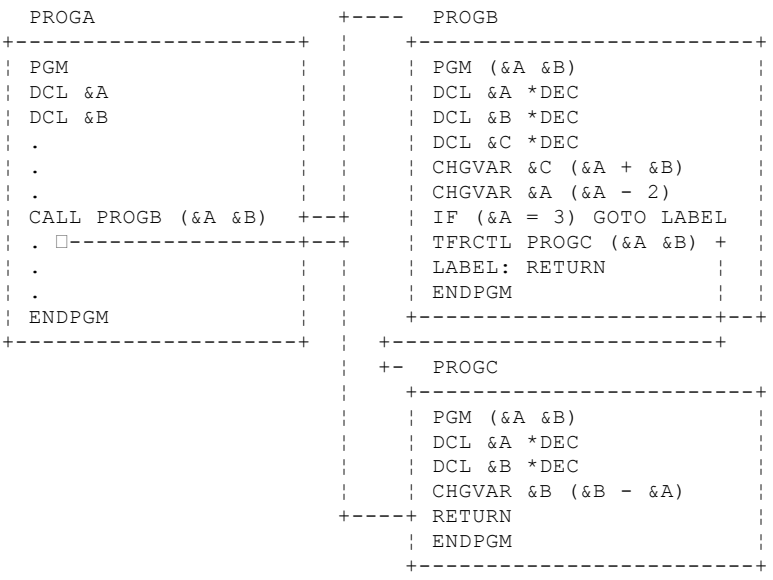
Los parámetros PARM se trata en el apartado "Paso de Parámetros entre Programas y Procedimientos" en el tema 3.4.

B.2 Paso de Parámetros

El mandato TFRCTL puede utilizarse para pasar parámetros al programa al que se llama de la misma forma en que el mandato CALL pasa parámetros, pero con las siguientes restricciones:

- Los parámetros pasados deben ser variables CL.
- El programa que transfiere el control debe haber recibido las variables CL pasadas como parámetros por otro programa.
- Este mandato sólo es válido en programas OPM CL.

En el ejemplo siguiente, PROGA llama a PROGB y le pasa dos variables, &A y &B. PROGB utiliza estas dos variables y otra variable declarada internamente, &C. Cuando se transfiere el control a PROGC, sólo se pueden pasar &A y &B a PROGC. Cuando PROGC termina el proceso, el control se devuelve a PROGA, donde se originaron estas variables.



C.0 Apéndice C. Archivos de Salida de Anotaciones de Trabajo

Subtemas

C.1 Direccionamiento de Anotaciones de Trabajo

Puede direccionar las anotaciones de trabajo para un trabajo a uno o dos archivos de base de datos con la API Controlar Anotaciones de Trabajo |(QMHCTLJL) API, o el mandato Visualizar Anotaciones de Trabajo |(DSPJOBLOG). El primer archivo es el archivo primario de anotaciones de trabajo. Este archivo contiene la información más importante para un mensaje, como ID de mensaje, tipo de mensaje y gravedad del mensaje. Se produce un registro en el archivo primario de las anotaciones de trabajo para cada mensaje seleccionado para procesar. El segundo archivo es el archivo secundario de anotaciones de trabajo. La producción de este archivo sólo es posible mediante la utilización de la API QMHCTLJL; sin embargo, también es opcional.

|El archivo secundario de anotaciones de trabajo contiene el texto de primer y segundo nivel para un mensaje. El texto está en formato de impresión. Todos los datos de mensaje se fusionan con la descripción del mensaje y el resultado se formatea en una o más líneas de impresión. Para cada mensaje seleccionado para procesar puede haber más de un registro en el archivo secundario de anotaciones de trabajo; un registro para cada línea de impresión de primer y segundo nivel.

Los registros del archivo primario pueden relacionarse con registros del archivo secundario mediante la Clave de Referencia de Mensajes. Cada registro del archivo primario contiene un campo que es la Clave de |Referencia de Mensajes (MRK) del mensaje relacionado. Igualmente, cada |registro de archivo secundario contiene la MRK del mensaje relacionado. La MRK para un mensaje es exclusiva dentro del contexto de un trabajo. Una vez conocida la MRK de un registro de archivo primario, los registros secundarios relacionados pueden identificarse fácilmente, ya que sólo estos registros secundarios contendrán el mismo valor de MRK.

C.2 Modelo para las Anotaciones de Trabajo Primarias

El modelo suministrado por IBM para el archivo primario de anotaciones de trabajo es QAMHJLPR en la biblioteca QSYS. El formato del registro primario es QMHPFT. A continuación se presenta una descripción detallada de este formato:

Orden Campo	**Nombre Campo**	**Tipo datos**	**Longitud en Bytes**	**Descripción Campo**
1	QMHJDT	DATE	10	Fecha creación anotaciones de trabajo
2	QMHJTM	TIME	8	Hora creación anotaciones de trabajo
3	QMHMRK	CHAR	4	Clave referencia mensajes
4	QMHSEV	BIN	4	Gravedad de mensaje
5	QMHTYP	CHAR	10	Tipo de mensaje
6	QMHMID	CHAR	7	ID de mensaje
7	QMHDAT	DATE	10	Fecha envío mensaje
8	QMHTIM	TIME	8	Hora envío mensaje
9	QMHMF	CHAR	20	Nombre de archivo de mensajes
10	QMHRPY	CHAR	4	Clave de referencia de respuesta
11	QMHRQS	CHAR	1	Estado Mensaje Petición
12	QMHSTY	CHAR	1	Tipo programa emisor
13	QMHRTY	CHAR	1	Tipo programa receptor
14	QMHSSN	BIN	4	Número sentencias para programa emisor
15	QMHRSN	BIN	4	Número sentencias para programa receptor
16	QMHCID	BIN	4	CCSID de datos de mensaje o mensaje inmediato
17	QMHPRL	CHAR	1	Indicador de mensaje transferido
18	QMHSPR	VAR CHAR	256 MAX	Nombre procedimiento emisor
19	QMHSMD	CHAR	10	Nombre módulo emisor
20	QMHSPG	CHAR	12	Nombre programa emisor
21	QMHSLB	CHAR	10	Nombre biblioteca emisora
22	QMHSTM	CHAR	30	Número sentencia(s) para programa emisor
23	QMHRPR	VAR CHAR	256 MAX	Nombre procedimiento receptor
24	QMHRMD	CHAR	10	Nombre módulo receptor
25	QMHRPG	CHAR	10	Nombre programa receptor
26	QMHRLB	CHAR	10	Nombre biblioteca programa receptor
27	QMHRTM	CHAR	30	Número sentencia(s) para programa receptor
28	QMHSYS	CHAR	8	Nombre del sistema
29	QMHJOB	CHAR	26	Nombre Trabajo Calificado
30	QMHMDT	VAR CHAR	3000 MAX	Datos de mensaje o mensajes inmediatos

	31	QMHCSP	VAR CHAR	4096 MAX	Nombre completo procedimiento emisor
	32	QMHCRP	VAR CHAR	4096 MAX	Nombre completo procedimiento receptor
	33	QMHLSP	VAR CHAR	6144 MAX	Largo del nombre del programa emisor

La definición de los campos en este registro es la siguiente:

QMHJDT
Fecha creación anotaciones de trabajo; DATE(10)
La fecha en la que empezó la producción de las anotaciones de trabajo. El campo es un campo de fecha en el registro de base de datos. El formato de la fecha es *ISO. Un valor en este campo de fecha está en el formato aaaa-mm-dd. Todos los registros producidos para las mismas anotaciones de trabajo tendrán el mismo valor en este campo.

QMHJTM
Hora de creación de las anotaciones de trabajo; TIME(8)
La hora en que empezó la producción de las anotaciones de trabajo. Este campo está definido como un campo de hora en el registro de base de datos. El formato de la hora es de tipo *ISO. Un valor en este campo de hora está en el formato hh.mm.ss. Todos los registros producidos para las mismas anotaciones de trabajo tendrán el mismo valor en este campo.

QMHMRK
Clave de referencia de mensaje; CHAR(4)
La clave de referencia de mensaje que tenía el mensaje relacionado en la cola de mensajes del trabajo. Los registros se colocan en el archivo de base de datos primario en estricto orden ascendente por clave de referencia. Dentro del conjunto de registros producidos por una sola anotación de trabajo, este campo es exclusivo para cada registro y por lo tanto puede utilizarse como clave exclusiva para el registro. Si los registros para dos o más anotaciones de trabajo se colocan en el mismo miembro, puede que la clave deje de ser exclusiva.

QMHSEV
Gravedad de mensaje; BIN(4)
La gravedad que tiene el mensaje. Este valor va de 0 a 99.

QMHTYP
Tipo mensaje; CHAR(10)
El tipo de mensaje del mensaje relacionado. En este campo aparecerá uno de los siguientes valores especiales.

***CMD**	Mandatos que se anotan desde la ejecución de un programa CL.
***COMP**	Tipo de mensaje de terminación.
***COPY**	Tipo de mensaje de copia del emisor.
***DIAG**	Tipo de mensaje de diagnóstico.
***ESCAPE**	Tipo de mensaje de escape.
***INFO**	Tipo de mensaje informativo.
***INQ**	Tipo de mensaje de consulta.
***NOTIFY**	Tipo de mensaje de notificación.
***RQS**	Tipo de mensaje de petición.
***RPY**	Tipo de mensaje de respuesta.

QMHMID
ID de mensaje; CHAR(7)
El ID del mensaje. Este campo contendrá el valor especial *IMMED si el mensaje es un mensaje inmediato que no tiene ID de mensaje.

QMHDAT
Fecha de envío del mensaje; DATE(10)
La fecha en que se envió el mensaje. Este campo está definido como un campo de fecha en el registro de base de datos. El formato de la fecha es *ISO. Un valor en este campo tiene el formato aaaa-mm-dd.

QMHTIM
Hora de envío del mensaje; TIME(8)
La hora en que se envió el mensaje. Este campo está definido como un campo de hora en el registro de base de datos. El formato de la hora es de tipo *ISO. El valor de este campo tiene el formato hh.mm.ss.

QMHMF
Archivo de mensajes; CHAR(20)
El nombre del archivo de mensajes que se utilizará para obtener la descripción del mensaje. Los 10 primeros caracteres del campo contienen el nombre de archivo de mensajes. Los segundos 10

caracteres contienen el nombre de la biblioteca. Si el campo QMHMID contiene *IMMED para indicar un mensaje inmediato, este campo contendrá todo blancos.

QMHRPY
Clave de referencia de respuesta; CHAR(4)

- Si el tipo de mensaje es de consulta, de notificación o de copia del emisor, esta es la clave de referencia de mensaje del mensaje de respuesta relacionado.
- Si no hay ningún mensaje de respuesta disponible, este campo contendrá un valor nulo ('00000000'X).
- Si tipo de mensaje no es de consulta, de notificación o de copia del emisor, este campo también contendrá un valor nulo.

Para mantener estrictamente el orden ascendente por clave de referencia, el registro para el mensaje de respuesta no puede ir inmediatamente después del registro para el mensaje de consulta, notificación o copia del emisor.

QMHRQS
Estado de Mensaje de Petición; CHAR(1)

- Si el tipo de mensaje es *RQS, este es un indicador que muestra si se ha ejecutado o no el mensaje de petición.
- Si el indicador está establecido a cero ('F0'X), no se ha ejecutado la petición.
- Si el indicador está establecido a uno ('F1'X), se ha ejecutado la petición.

Si el tipo de los mensajes no es *RQS, este indicador siempre será cero.

QMHSTY
Tipo de programa emisor; CHAR(1)
Un indicador con los siguientes valores que indica si el programa emisor era un programa OPM o un programa ILE.

- Si el indicador está establecido a cero ('F0'X), el programa emisor es un OPM o un programa de Código Interno bajo Licencia (SLIC) con un nombre igual o inferior a 12 caracteres. El nombre del programa se sitúa en los campos QMHSPG y QMHLSP.
- Si el indicador está establecido en uno ('F1'X), el programa emisor es un programa ILE con un nombre de procedimiento menor o igual a 256 caracteres. El nombre del procedimiento se sitúa en los campos QMHSPR y QMHCSP.
- Si el indicador está establecido a dos ('F2'X), el programa emisor es un programa ILE con un nombre de procedimiento mayor de 256 y hasta 4096 caracteres. El nombre del procedimiento emisor completo se encuentra en el campo QMHCSP; el campo QMHSPR está en blanco.
- Si el indicador está establecido a tres ('F3'X), el programa emisor es un programa SLIC con un nombre de procedimiento mayor de 12 y hasta 256 caracteres. El nombre del procedimiento emisor completo se encuentra en el campo QMHLSP; el campo QMHSPG está en blanco.

QMHRTY
Tipo de programa receptor; CHAR(1)
Un indicador con los siguientes valores que indica el tipo de programa receptor:

- Si el indicador está establecido a cero ('F0'X), el programa receptor era un programa OPM. El nombre del programa se sitúa en el campo QMHRPG.
- Si el indicador está establecido a uno ('F1'X), el programa receptor era un programa ILE con un nombre de procedimiento menor o igual a 256 caracteres. El nombre del procedimiento se sitúa en los campos QMHRPR y QMHCRP.
- Si el indicador está establecido a dos ('F2'X), el programa receptor es un programa ILE con un nombre de procedimiento mayor de 256 y hasta 4096 caracteres. El nombre completo del procedimiento receptor se sitúa en el campo QMHCRP; el campo QMHRPR está en blanco.

QMHSSN
Número de sentencias para el programa emisor; BIN(4)
La cantidad números de sentencia para el programa emisor.

- Si el campo de tipo de programa emisor QMHSTY contiene un cero ('F0'X) o un tres ('F3'X), este campo contiene un valor de 0 ó 1.
- Si el campo de tipo de programa emisor contiene un uno ('F1'X) o un dos ('F2'X), este campo puede contener los valores 0, 1, 2, ó 3.

| El valor proporcionado en este campo define cuántos números de
| sentencia hay en el campo QMHSTM.

QMHRSN
Número de sentencias para el programa receptor; BIN(4)
El número de sentencias para el programa receptor.

| - Si el campo de tipo de programa receptor QMHRTY contiene un cero
| ('F0'X), este campo contiene un valor de 0 a 1.
| - Si el campo tipo de programa receptor contiene un uno ('F1'X) o un
| dos ('F2'X), este campo contiene los valores 0, 1, 2 ó 3. El
| valor proporcionado en este campo define cuántos números de
| sentencia hay en el campo QMHRTM.

QMHCID
CCSID de los datos del mensaje; BIN(4)
El CCSID de los datos del mensaje o mensaje inmediato contenidos en el campo QMHMDT.

QMHPRL
Indicador de si se ha transferido el mensaje; CHAR(1)
Un indicador que muestra si el mensaje se transfirió al programa receptor o no.

| - Si el mensaje no se transfirió, este campo contiene un cero
| ('F0'X).
| - Si el mensaje se envió, este campo contiene un uno ('F1'X).

| La transferencia de mensajes sólo puede producirse dentro de un
| programa ILE. Por consiguiente, este campo contiene un uno sólamente
| si el campo de tipo de programa receptor QMHRTY contiene un uno
| ('F1'X) o un dos ('F2'X).

QMHSPR
Nombre del procedimiento emisor; VAR CHAR(*)

| - Si el campo de tipo de programa emisor QMHSTY contiene un cero
| ('F0'X) o un tres ('F3'X), este campo contiene el valor *N.
| - Si el campo tipo de programa emisor QMHSTY contiene un uno
| ('F1'X), este campo contiene el nombre del procedimiento ILE
| emisor. El nombre puede tener 256 caracteres como máximo.
| - Si el campo de tipo de programa emisor QMHSTY contiene un dos
| ('F2'X), este campo contiene blancos, mientras que el nombre
| completo se contendrá en el campo QMHCSP.

| Este campo puede contener un nombre de procedimieno jerarquizado para
| un programa emisor de tipo uno ('F1'X) o dos ('F2'X); cada nombre de
| procedimiento se separa con dos puntos. El nombre del procedimiento
| más externo se identifica en primer lugar, y va seguido por los
| procedimientos que contiene. Los procedimientos más internos, se
| identifican en último lugar en la serie.

|**QMHSMD**
| Nombre de módulo emisor; CHAR(10)

| - Si el campo de tipo de programa emisor QMHSTY contiene un cero
| ('F0'X) o un tres ('F3'X), este campo contiene el valor *N.
| - Si el campo de tipo de programa emisor QMHSTY contiene un uno
| ('F1'X) o un dos ('F2'X), este campo contiene el nombre del módulo
| ILE emisor.

|**QMHSPG**
| Nombre de programa emisor; CHAR(12)

| - Si el campo de tipo de programa emisor QMHSTY contiene un cero
| ('F0'X), un uno ('F1'X), o un dos ('F2'X), el campo contiene el
| nombre del programa desde el que se envió el mensaje.
| - Si el tipo de programa emisor es un tres ('F3'X), este campo
| contiene blancos, y el campo QMHLSP contiene el nombre del
| programa emisor.

QMHSLB
Nombre de biblioteca emisora; CHAR(10)
El nombre de la biblioteca que contenía el programa emisor.

|**QMHSTM**
| Número(s) de sentencia para el programa emisor; CHAR(30)
| Número(s) de sentencia en la que el programa emisor envió el mensaje.
| Cada número de sentencia tiene 10 caracteres de longitud.

| - Si el campo tipo de programa emisor QMHSTY contiene un cero
| ('F0'X) o un tres (F3'X), hay como máximo, un número de sentencia
| en los 10 primeros caracteres. Este número de sentencia

| representa un número de instrucción MI. El número es hexadecimal.
| - Si el campo de tipo de programa emisor contiene un uno ('F1'X) o un dos ('F2'X), este campo puede contener los números de sentencia 0, 1, 2, ó 3. El campo QMHSSN especifica cuántos hay. En este caso, un número de sentencia es un número de sentencia de lenguaje de nivel superior, y no un número de instrucción MI. Cada número es decimal.

QMHRPR
Nombre del procedimiento receptor; VAR CHAR(*)

| - Si el campo de tipo de programa receptor contiene un cero ('F0'X), este campo contiene el valor *N.
- Si el campo tipo de programa receptor QMHRTY contiene un uno ('F1'X), este campo contiene el nombre del procedimiento ILE receptor. El nombre puede tener 256 caracteres como máximo.
- Si el campo de tipo de programa receptor QMHRTY contiene un dos ('F2'X), este campo contiene blancos, mientras que el nombre completo estará contenido en el campo QMHCRP.

Este campo puede contener un nombre de procedimiento jerarquizado para un tipo de programa emisor 1 ó 2; cada nombre de procedimiento se
| separa con dos puntos. El nombre del procedimiento más externo se
| identifica en primer lugar, y va seguido por los procedimientos que
| contiene. Los procedimientos más internos, se identifican en último lugar en la serie.

|**QMHRMD**
| Nombre del módulo receptor; CHAR(10)

| - Si el campo de tipo de programa receptor contiene un cero ('0F'X'), este campo contiene el valor *N.
| - Si el campo de tipo de programa receptor QMHRTY contiene un uno ('F1'X) o un dos ('F2'X), este campo contiene el nombre del módulo ILE receptor.

QMHRPG
Nombre del programa receptor; CHAR(10)
El nombre de programa OPM o ILE al que se envió el mensaje.

QMHRLB
Nombre de la biblioteca receptora; CHAR(10)
El nombre de la biblioteca que contenía el programa receptor.

|**QMHRTM**
| Número(s) de sentencia para el programa receptor; CHAR(30)
| Número(s) de sentencia en que se detuvo el programa receptor cuando se envió el mensaje. Cada número de sentencia tiene 10 caracteres de longitud.

| - Si el campo de tipo de programa receptor QMHRTY contiene un cero ('F0'X), hay, como máximo, un número de sentencia en los primeros 10 caracteres. Este número de sentencia representa un número de instrucción MI. El número es hexadecimal.

| Para cualquier otro valor del tipo de programa receptor, en este campo pueden haber 0, 1, 2, ó 3 números de sentencia. El campo QMHRSN
| especifica cuántos hay. En este caso, un número de sentencia es un número de sentencia de lenguaje de nivel superior, y no un número de instrucción MI. Cada número es decimal.

QMHSYS
Nombre del sistema; CHAR(8)
El nombre del sistema en el que se produjeron las anotaciones de trabajo.

QMHJOB
Nombre calificado de trabajo; CHAR(26)
El nombre totalmente calificado del trabajo para el que se está anotando el mensaje. Las 10 primeras posiciones contienen el nombre de trabajo, las siguientes 10 posiciones el nombre de usuario, y las 6 últimas posiciones contienen el número de trabajo.

QMHMDT
Datos del mensaje; VAR CHAR(*)
Si el campo QMHMID contiene el valor especial *IMMED, este campo contiene un mensaje inmediato. De lo contrario, este campo contiene datos utilizados cuando se envió el mensaje. Este campo puede contener un máximo de 3000 caracteres. Si el mensaje inmediato o los datos son más largos, éstos se truncan a 3000 caracteres.

Si los datos del mensaje contienen punteros, éstos se invalidan antes de que los datos del mensaje se graben en el archivo de base de datos.

QMHCSP
Nombre completo del procedimiento emisor; CHAR(VAR)

- Si el tipo de programa emisor es cero ('F0'X) o tres ('F3'X), este campo contiene blancos.
- Si el tipo de programa emisor es uno ('F1'X) o dos ('F2'X), este campo contiene el nombre completo del procedimiento ILE. El nombre puede tener 4096 caracteres como máximo.

Este campo puede contener un nombre de procedimiento jerarquizado donde cada nombre de procedimiento se separa con dos puntos. El nombre del procedimiento más externo se identifica en primer lugar, y va seguido por los procedimientos que contiene. Los procedimientos más internos, se identifican en último lugar en la serie.

QMHCRP
Nombre completo del procedimiento receptor; CHAR(VAR)

- Si el tipo de programa receptor es cero ('F0'X), este campo contiene blancos.
- Si el tipo de programa receptor es uno ('F1'X) o dos ('F2'X), este campo contiene el nombre completo del procedimiento ILE. El nombre puede tener 4096 caracteres como máximo.

El campo puede contener un nombre de procedimiento jerarquizado donde cada nombre de procedimiento se separa con dos puntos. El nombre del procedimiento más externo se identifica en primer lugar, y va seguido por los procedimientos que contiene. Los procedimientos más internos, se identifican en último lugar en la serie.

QMHLSP
Nombre largo del programa emisor; CHAR(VAR)
Este campo contiene, para todos los tipos de programas emisores, el nombre completo del programa emisor desde el que se envió el mensaje. El nombre puede tener 6144 caracteres como máximo.

El modelo suministrado por IBM para el archivo de anotaciones de trabajo secundarias es QAMHJLSC de la biblioteca QSYS. El nombre de formato de registro secundario es QMHSFT. A continuación se presenta una descripción detallada del formato de registro secundario:

Orden Campo	Nombre Campo	Tipo datos	Longitud en Bytes	Descripción Campo
1	QMHJDT	DATE	10	Fecha creación anotaciones de trabajo
2	QMHJTS	TIME	8	Hora creación anotaciones de trabajo
3	QMHMKS	CHAR	4	Clave referencia mensajes
7	QMHSYN	CHAR	8	Nombre del sistema
8	QMHJBN	CHAR	26	Nombre Calificado Trabajo
4	QMHLNN	BIN	4	Número línea mensaje
5	QMHSID	BIN	4	CCSID de línea de texto
6	QMHTTY	CHAR	1	Indicador texto mensaje
9	QMHLIN	CHAR	78	Línea texto mensaje

La longitud del campo indica el número total de bytes para el campo.

La definición de los campos en este registro es la siguiente:

QMHJDT
Fecha creación anotaciones de trabajo; DATE(8)
La fecha en la que empezó la producción de las anotaciones de trabajo. El campo es un campo de fecha en el registro de base de datos. El formato de la fecha es *ISO. Un valor en este campo tiene el formato aaaa-mm-dd. Todos los registros producidos para las mismas anotaciones de trabajo tendrán el mismo valor en este campo.

QMHJTS
Hora creación anotaciones de trabajo; TIME(8);
La hora en que empezó la producción de las anotaciones de trabajo.

Este campo está definido como un campo de hora en el registro de base de datos. El formato de la hora es de tipo *ISO. El valor de este campo tiene el formato hh.mm.ss. Todos los registros producidos para las mismas anotaciones de trabajo tendrán el mismo valor en este campo.

QMHMKS
Clave de referencia de mensaje; CHAR(4)
La clave de referencia de mensaje que tenía el mensaje relacionado en la cola de mensajes del trabajo. Los registros se colocan en el archivo de base de datos secundario en estricto orden ascendente por clave de referencia. Puede haber más de un registro secundario para una clave de referencia de mensaje específica. Este campo también existe en el registro primario relacionado. Por consiguiente, una vez obtenida la clave de referencia de mensaje desde un registro primario, puede utilizarse para leer los registros relacionados del archivo secundario.

QMHSYN
Nombre del sistema; CHAR(8)
El nombre del sistema en el que se produjeron las anotaciones de trabajo.

QMHJBN
Nombre calificado de trabajo; CHAR(26)
El nombre totalmente calificado del trabajo para el que se está anotando el mensaje. Las 10 primeras posiciones contienen el nombre de trabajo, las siguientes 10 posiciones el nombre de usuario, y las 6 últimas posiciones contienen el número de trabajo.

QMHLNN
Número línea mensaje; BIN(4)
El número de línea de la línea dentro del tipo de texto. Para el texto de primer y segundo nivel, el número de línea empieza en uno para la primera línea del texto y se incremente en uno para cada línea adicional dentro de ese nivel.

QMHSID
CCSID de la línea de texto del mensaje; BIN(4)
El CCSID de la línea de texto del mensaje contenida en el campo QMHLIN.

QMHTTY
Tipo de texto del mensaje; CHAR(1)
Un indicador que especifica si el campo QMHLIN contiene una línea de texto de primer o segundo nivel. Este campo contendrá uno de los siguientes valores:

1 El campo QMHLIN contiene texto de primer nivel.
2 El campo QMHLIN contiene texto de segundo nivel.

QMHLIN
Línea de texto del mensaje: CHAR(78)
Este campo contiene una línea de texto de primer o segundo nivel.

BIBLIOGRAFIA Bibliografía
A continuación se facilita una lista de los libros adicionales y una descripción de la información de cada libro.

Para obtener información sobre el funcionamiento del sistema AS/400 y sus estaciones de pantalla, véase:

- *Operación del Sistema para Nuevos Usuarios*, SC10-9268 (SC41-3200)

  Esta guía proporciona información general sobre la utilización de estaciones de pantalla, e incluye teclas de función, pantallas, mandatos e información de ayuda.

- *Operación del sistema*, SC10-9621 (SC41-4203)

  Esta guía proporciona información general sobre cómo ejecutar el sistema, cómo enviar y recibir mensajes, así como sobre la utilización de las teclas de función de la estación de pantalla.

Para más información sobre la programación del AS/400, consulte:

- Application Display Programming, SC41-4715

  Esta guía proporciona información sobre cómo utilizar las DDS para crear y mantener pantallas, crear y trabajar con archivos de pantalla, crear información de ayuda en línea, utilizar las UIM para definir pantallas y utilizar grupos de panel, registros y documentos.

- Backup and Recovery - Advanced, SC41-4305

  Esta guía proporciona información sobre los distintos soportes de almacenamiento disponibles para salvar y proteger los datos del sistema.

- *Gestión de datos*, SC10-9635 (SC41-4710)

  Esta guía proporciona información sobre la estructura y conceptos del soporte de gestión de datos, alteraciones temporales y redirección de archivos, copia de archivos y adaptación de un sistema que utiliza datos de doble byte.

- *DB2/400 Programación de la base de datos*, SC10-9634 (SC41-4701)

  Esta guía proporciona información detallada de la estructura de la base de datos del AS/400, que incluye información sobre cómo crear, describir y manipular archivos de base de datos.

- DDS Reference, SC41-3712

  Este libro proporciona una descripción detallada de las entradas y palabras clave necesarias para describir externamente archivos de bases de datos y determinados archivos de dispositivo.

- *ILE Conceptos*, SC10-9631 (SC41-4606)

  Este libro explica conceptos y terminología relativa a la arquitectura de Entorno de Lenguajes Integrados (ILE) del programa bajo licencia OS/400. Los temas tratados en este manual incluyen la creación de módulos, el enlace y ejecución de programas, la depuración de programas y el manejo de excepciones.

- National Language Support, SC41-3101

  Esta guía proporciona al gestor de proceso de datos, gestor y operador del sistema, programador de aplicaciones, usuario final, representante de ventas de IBM e ingeniero de aplicaciones información necesaria para comprender y utilizar la función de soporte de idioma nacional en el sistema AS/400. Este libro prepara al usuario del AS/400 para planificar, instalar, configurar y utilizar el soporte de idiomas nacionales (NLS) AS/400 y el soporte plurilingüe del sistema AS/400. Asimismo, proporciona una explicación sobre la gestión de bases de datos multilingües y consideraciones sobre aplicaciones para un sistema multilingüe.

- International Application Development, SC41-3603

  Esta guía proporciona al gestor de proceso de datos, gestor y operador del sistema, programador de aplicaciones, usuario final, representante de ventas de IBM e ingeniero de aplicaciones información necesaria para comprender y utilizar la función de soporte de idioma nacional en el sistema AS/400. Este libro prepara al usuario del AS/400 para planificar, instalar, configurar y utilizar el soporte de idiomas nacionales (NLS) AS/400 y el soporte plurilingüe del sistema AS/400.

Asimismo, proporciona una explicación sobre la gestión de bases de datos multilingües y consideraciones sobre aplicaciones para un sistema multilingüe.

- Printer Device Programming, SC41-3713

  Esta guía proporciona información específica sobre elementos de impresión y conceptos del sistema AS/400, archivo de impresora y soporte al spooling de impresión y conectividad de impresora.

- *Programación - Resumen de consulta*, SX10-8531 (SX41-4720)

  Esta publicación proporciona acceso rápido y fácil a información resumen referente a muchos de los lenguajes y programas de utilidad disponibles en el sistema AS/400.

- Security - Reference, SC41-4302

  Este manual explica conceptos generales de seguridad y planificación de la seguridad en el sistema. También incluye información para todos los usuarios acerca de la seguridad de los recursos.

- Tape and Diskette Device Programming, SC41-4716

  Esta guía proporciona información para ayudar a los usuarios a desarrollar y dar soporte a programas que utilizan unidades de cinta y de disquetes para E/S. Incluye información sobre los archivos de dispositivo y descripciones para los dispositivos de cinta y de disquetes, así como spooling para dispositivos de disquetes.

- *Gestión de Trabajos*, SC10-9627 (SC41-4306)

  Esta guía proporciona información sobre la creación y la modificación del entorno de gestión de trabajo, el trabajo con valores del sistema y la recogida y el uso de datos de rendimiento para aumentar el rendimiento del sistema.

Para ver información detallada referente a APIs, consulte:

- System API Programming, SC41-3800

  Este manual proporciona información para programadores expertos que desean utilizar las interfaces de programación de aplicaciones (APIs) del OS/400. También proporciona ejemplos a título introductorio para ayudar al programador en la utilización de las API.

- System API Reference, SC41-4801

  Esta publicación ofrece información para el programador experto referente a cómo utilizar las interfaces de programación de aplicaciones (APIs) para funciones OS/400. También incluye APIs OPM (modelo programa original), ILE (Entorno de Lenguajes Integrados) y estilo UNIX.

Para más información sobre los programas de utilidad del AS/400 mencionados en este manual, consulte:

- ADTS/400: Character Generator Utility

  Esta guía proporciona información sobre el uso del programa de utilidad del generador de caracteres (CGU) para crear y mantener un juego de caracteres de doble byte en el sistema AS/400.

- *ADTS/400: Gestor de Desarrollo de Programas (PDM)*

  Esta publicación proporciona información referente a la utilización del Gestor para el Desarrollo de Programación (PDM) para trabajar con listas de bibliotecas, objetos, miembros y opciones definidas por el usuario.

- *ADTS/400: Ayuda para el Diseño de Pantallas (SDA)*

  Este libro proporciona información acerca de la utilización de la Ayuda para el Diseño de Pantallas (SDA) para diseñar, crear y mantener formatos de pantalla y menús.

- *ADTS/400: Programa de Utilidad para Entrada del Fuente (SEU)*

  Este libro proporciona información acerca de la utilización del Programa de Utilidad para Entrada del Fuente (SEU) para crear y editar miembros fuente.

Para obtener más información acerca de RPG Multimedia Tutorial, consulte:

- Experience RPG IV Multimedia Tutorial

  Esto es un programa de autoaprendizaje interactivo que explica las diferencias entre RPG III y RPG IV y cómo trabajar en el nuevo entorno ILE. Un libro de ejercicios suplementario proporciona ejercicios adicionales y al mismo tiempo sirve de consulta a lo largo de esta guía. Con la guía se proporcionan ejemplos de código ILE RPG/400 y se ejecutan directamente en el AS/400. Llame al 1-800-IBM-CALL para efectuar el pedido de la guía.

B

D

E

P

W

COMENTARIOS Hoja de Comentarios
AS/400 Advanced Series
CL Programación
Versión 3 Release 6

Número de Publicación SC10-9637-00

En general, ¿está Ud. satisfecho con la información de este libro?

Valores:

1 Muy satisfecho
2 Satisfecho
3 Normal
4 Insatisfecho
5 Muy insatisfecho

	1	2	3	4	5
Satisfacción general					

¿Cómo valora los siguientes aspectos de este libro?

	1	2	3	4	5
Organización					
Información completa y precisa					
Información fácil de encontrar					
Utilidad de las ilustraciones					
Claridad de la redacción					
Calidad de la edición					
Adaptación a los formatos, unidades, etc. del país					

Comentarios y sugerencias:

IBM, S.A.
National Language Solutions Center
Av. Diagonal, 571
E-08029 Barcelona

Nombre . . . . . _____
Compañía u Organización _____
Dirección . . . . . _____
_____
_____
Teléfono . . . . . _____