JCL (Job Control Language)
MVS/ESA (Multiple Virtual Storage/Enterprise System Architecture)

Marzo de 1999

Material para el participante

TABLA DE CONTENIDOS

Prefacio

Este documento tiene como finalidad describir las sentencias de JCL más importantes que permitiran crear JOBs capaces de correr en el host MVS.
Estas sentencias, permitirán la comunicación en forma batch con el sistema operativo MVS (Multiple Virtual Storage).
Es importante destacar que en este documento figuran las sentencias fundamentales y sus parámetros más usuales.

Introducción

El Job Control Language (JCL) es el medio empleado para comunicarse con el Sistema Operativo y Job Entry Subsystem (JES2).

Para ello, se emplean JOBs (uno o varios grupos de sentencias de control) que informan al sistema los programas a ejecutar, los archivos que estos usaran, la cantidad de memoria necesaria, las caracteristicas de la tarea, el tipo de salida, etc.

Un JOB puede estar formado por uno o varios pasos (job steps), cada uno de los cuales es una unidad de trabajo asociada a un programa. Un JOB puede tener un máximo de 255 steps.

Relación del JCL con JES2

Todo JOB es tratado por el Job Entry Subsystem (JES2), como un archivo ordinario, es decir, es leído y puesto en una cola de despacho que luego será atendida de acuerdo a la carga de trabajo del sistema. Asimismo las salidas impresas generadas por el JOB, serán administradas por JES2.

Sentencias de JCL

El JCL es un conjunto de sentencias, las cuales se resumen a continuación:

Sentencia	Propósito
// JOB	Marcar el comienzo de un trabajo y asignarle un nombre.
// EXEC	Marcar el comienzo de un paso y asignarle un nombre, identificar al programa o procedimiento a ejecutar.
// DD	Identificar y describir un archivo.
// PROC	Marcar el comienzo de un procedimiento, asignar valores por defecto a los parámetros definidos en el procedimiento.
// PEND	Marcar el final de un procedimiento.
//*	Contener comentarios.
/*	Indicar el final de ingreso de datos.
//	Marcar el final de un trabajo.

La sintaxis genral de las sentencias de control es la siguiente:

//	Dos barras en columnas 1 y 2. Un rótulo de hasta 8 caracteres alfanumericos comenzando con letra entre las columnas 3 y 10. El nombre de la sentencia dejando por lo menos un blanco luego del rótulo y no despues de la columna 16. Los parámetros separados por coma, dejando por lo menos un blanco despues del nombre de sentencia.

Se podrá escribir hasta la columna 71, y de no ser suficiente para colocar todos los parámetros necesarios, se colocará una coma(,) como último caracter y se continuará en líneas susecivas comenzando con // en columnas 1 y 2 y escribiendo a partir de las columnas 3 a 16.

Sentencia JOB

Esta sentencia dede ser la primera en cada conjunto de sentencias y le indica al sistema como procesar el trabajo. No puede aparecer dentro de un procedimiento.

SINTAXIS

```
//jobname JOB parámetros_posicionales,parámetros_de_palabra_clave
```

Tipos de Parámetros

Existen dos tipos de parámetros:

Posicionales: hay dos parámetros de este tipo, los que deben codificarse en el siguiente orden:

1 - Información de **accounting** o contabilidad que permite sumarizar los recursos utilizados.
2 - Nombre del responsable (opcional).

Palabra clave: pueden ser codificados en cualquier orden, luego de los posicionales.

1 - **CLASS=**	Indica la clase de ejecución del JOB. En general es una letra o un número y depende de la instalación.
2 - **COND=**	Establece las condiciones para la ejecución. Si por ejemplo el JOB anterior finalizó con un código de condición distinto de 0, este no se ejecutara (0,NE).
3 - **MSGCLASS=**	Define la clase de salida de los mensajes. En general es una letra o un número dependiendo de la instalación.
4 - **MSGLEVEL=**	Selecciona el tipo de mensajes a emitir. Por ejemplo puede desearse que el sistema no emita mensajes de diagnóstico de la ejecución si no hubo errores, o que los emita aun cuando no hubiera errores. Ej. (1,1).
5 - **NOTIFY=**	Indica el nombre del usuario que recibirá los mensajes.
6 - **REGION=**	Establece la cantidad de memoria a utilizar. Ej. 800K
7 - **TIME=**	Determina el tiempo máximo de uso de CPU que durará el JOB. Asume minutos. Ej. (20)
8 - **RESTART=**	Marca el paso donde comenzará el JOB. Si no se desea arrancar desde el comienzo, se indicará el nombre del paso. Ej. PASO01
9 - **TYPRUN=**	Selecciona el tipo de ejecución.

HOLD - El JOB queda en la cola de entrada sin ejecutarse hasta que el operador lo libere.
SCAN - El JOB no se ejecuta, sólo se verifica la sintaxis de las sentencias.

Ejemplo:

```
//COMPCOB  JOB (023122),'GARCIA',CLASS=A,NOTIFY=USR005,
//   MSGCLASS=X,TIME=(2),REGION=2M
```

Sentencia EXEC

Identifica el programa o procedimiento que se ejecutará en este paso e indica al sistema como procesarlo. Debe ser la primer sentencia de cada paso o procedimiento.

SINTAXIS

```
//stepname EXEC parámetros_posicionales,parámetros_de_palabra_clave
```

Existen dos tipos de parámetros:

Posicionales: hay dos parámetros de este tipo, los que deben codificarse en el siguiente orden:

1 -	PGM=	Nombre del programa a ejecutar.
2 -	PROC=	Nombre del procedimiento a jecutar.
3 -	procname	Nombre del procedimiento a ejecutar.

Palabra clave: pueden ser codificados en cualquier orden, luego de los posicionales.

1 - **COND=**	Establece las condiciones para la ejecución. Si por ejemplo el paso anterior finalizó con un código de condición distinto de 0, este no se ejecutara (0,NE).
2 - **REGION=**	Establece la cantidad de memoria a utilizar. Ej. 800K
3 - **TIME=**	Determina el tiempo máximo de uso de CPU que durará el PASO.Asume minutos. Ej. (20). La suma de los TIME de cada paso no puede exceder el TIME establecido en la sentencia JOB.
4 - **PARM=**	lindicará los parámetros esperados por el programa o por el procedimiento.

Ejemplos:

```
//PASO01  EXEC PGM=IEBCOPY

//PASO02  EXEC COMPILAR          o bien
//PASO02  EXEC PROC=COMPILAR

//PASO03  EXEC PGM=IEWL,PARM='XREF,LIST,MAP',COND=(4,GT)
```

Sentencia DD

La sentencia DD (Dataset Definition), define los archivos a utilizar en un JOB o procedimiento, y especifica las características de los archivos sean estos de entrada o de salida. Deberá colocarse inmediatamente a continuación de la sentencia EXEC.
Existen algunas DD especiales, tales como la //JOBLIB, que se coloca luego de la sentencia JOB y define una o más bibliotecas que seran válidas para todos los pasos del JOB.
La //STEPLIB que define una o más bibliotecas válidas para el paso donde se ubica.

SINTAXIS

//ddname **DD** parámetros_posicionales,parámetros_de_palabra_clave

Existen dos tipos de parámetros:

Posicionales: cada sentencia puede tener solamente uno de los siguientes:

1 -	*	Indica que los datos del archivo vienen a continuación.
2 -	DUMMY	Indica que el archivo es simulado (no existe).

Palabra clave: pueden ser codificados en cualquier orden, luego de los posicionales (si estos existen).

1 - **DSN=**	Data Set Name o nombre del archivo físico. De hasta 44 caracteres alfanuméricos divididos en grupos de hasta 8 y separados por puntos, comenzando cada grupo con letra.
2 - **DISP=**	Establece la disposición del archivo: SHR De uso compartido. (Generalmente para lectura) OLD De uso exclusivo. (Para grabar encima de los datos que contenga) MOD De uso exclusivo. (Para grabar a continuación de lo existente) NEW De uso exclusivo. (Para crear un archivo nuevo) Especifica como quedará el archivo luego de usado si el paso finaliza correctamente: CATLG Quedará registrado en el catálogo. KEEP Será mantenido como estaba. PASS Será mantenido para ser usado por otro paso posterior. DELETE Será borrado.

Indica como quedará el archivo si el paso finaliza con errores:
Idem anterior.

3 - **DCB=** Define las características del archivo:

LRECL	Longitud de registro.	
BLKSIZE	Tamaño del bloque.	
RECFM	Formato del registro:	
	F	De longitud Fija.
	V	De longitud variable.
	FB	De longitud fija, bloqueado.
	PO	Particionado.

4 - **UNIT=** Determina el tipo de dispositivo físico donde se almacenará el archivo. Puede ser genérico o especifico.

5 - **SPACE=** Determina la cantidad de espacio físico que ocupará.
Puede expresarse en Registros, Pistas o Cilindros y puede asignarse una cantidad primaria y una alternativa.

6 - **VOL=SER=** Indica el nombre del volumen donde se alojará el archivo.

Ejemplos:

```
//SALIDA    DD DSN=ARCHIVO.DE.PRUEBA.CURSO,
//             DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(5,2)),
//             UNIT=3390,VOL=SER=DISCO1,
//             DCB=(LRECL=80,RECFM=FB,BLKSIZE=3120)
//ENTRADA DD DSN=MAESTRO.CLIENTES,DISP=SHR
//SYSIN    DD *
//SYSOUT  DD SYSOUT=X,COPIES=2
```

Sentencia PROC

Esta sentencia tiene por objeto indicar el comienzo de un procedimiento catalogado o "in-stream".

SINTAXIS

```
//nombre PROC [parámetros opcionales]
```

Los parámetros son opcionales y se emplean para permitir ejecutar el mismo procedimiento en distimtas circunstancias. Asi por ejemplo, pueden definirse nombres de archivos con variables simbólicas uqe al invocar al procedimiento para su ejecución podrán reemplazarse por los nombres reales de los archivos a usar en esa corrida. Los valores que se asignan a estas variables en el procedimiento, son los valores "default" o por defecto. Si al invocar al procedimiento, no se especifican parámetros, el procedimiento correrá con estos valores.

Ejemplo:

```
//COPIA  PROC ENTRADA='ARCHIVO.MAESTRO',
//       SALIDA='ARCHIVO.MAESTRO.BACKUP',
//       DISEN='SHR',DISSA='OLD'
//PASO1  EXEC PGM=IEBCOPY
//SYSUT1 DD DSN=&ENTRADA,DISP=&DISEN
//SYSUT2 DD DSN=&SALIDA,DISP=&DISSA
//SYSPRINT DD SYSOUT=X
//       PEND
```

Cuando este procedimiento se ejecute, podrá hacerse de la siguiente manera:

```
//BACKUP JOB (123456),CLASS=A,NOTIFY=OPER03
//JS001   EXEC COPIA,ENTRADA='ARCHIVO.CLIENTES',
//        SALIDA='COPIA.CLIENTES'
```

El JOB resultante, quedará armado de la siguiente forma:

```
//BACKUP JOB (123456),CLASS=A,NOTIFY=OPER03
//PASO1  EXEC PGM=IEBCOPY
//SYSUT1 DD DSN=ARCHIVO.CLIENTES,DISP=SHR
//SYSUT2 DD DSN=COPIA.CLIENTES,DISP=OLD
//SYSPRINT DD SYSOUT=X
```

Sentencia PEND

Su función es indicar el final de un procedimiento. No lleva parámetros.

SINTAXIS

```
//        PEND
```

Sentencia // (Comentario)*

Permite colocar comentarios en cualquier ubicación (luego de la sentencia JOB). Puede haber varias en un JOB y en distintas ubicaciones. Se utiliza para resaltar alguna situación en particular o bien como medio de documentación de JOBs muy extensos.

SINTAXIS

```
//*     comentarios
```

Ejemplo:

```
//* ****************************************
//* * EL SIGUIENTE PASO HACE UNA COPIA*
//* *       DEL ARCHIVO DE CLIENTES        *
//* ****************************************
```

Sentencia / (Fin de datos)*

Esta sentencia se usa para indicar el fin de datos, cuando estos estan incluidos como parte del JCL. No posee parámetros y debe estar en blanco hasta la columna 80.

SINTAXIS

```
/*
```

Ejemplo:

```
//LECTURA  JOB (123456),CLASS=A,NOTIFY=OPER03
//PASO1   EXEC PGM=IEBCOPY
//SYSUT2  DD  DSN=ARCHIVO.DE.DATOS,DISP=OLD
//SYSUT1  DD  *
876     LOPEZ, JOSE       MAGALLANES 234        CAP.FED.
225     PARDO, RENE       SAN MARTIN 1298       PALOMAR
756     CALVO, MIGUEL     ATUCHA 19             CASTELAR
/*
//SYSPRINT  DD  SYSOUT=X
```

Sentencia // (Fin de JOB)

Indica el final de un JOB. Si no se codifica, el sistema reconoce el comienzo del JOB siguiente. En caso de colocarse en medio de un JOB, este finalizará en ese punto, ignorando el resto de las sentencias de control. No posee parámetros y debe estar en blanco hasta la columna 80.

SINTAXIS

```
//
```