

CLIST (II)

0.- Entradilla

Este mes se continua tratando un lenguaje que es muy especial en todos los Grandes Sistemas, pues, por una parte, tiene la particularidad de estar instalado en todos los Centros de Procesos de Datos, pero por otra, resulta ser el gran desconocido de los programadores de estas instalaciones.

1.- Utilidad JMP01

El pasado mes se dedicó esta sección a tratar las generalidades de este lenguaje de comandos y de cómo se podía invocar un procedimiento. A modo de ejemplo se acompañó el artículo con una utilidad. Este mes se retoma el tema comenzando por comentar las distintas sentencias que componen dicha utilidad, pues su autor ha procurado que contenga el set de instrucciones básicas que componen cualquier lenguaje de programación tales como comentarios, sentencias de asignación, bifurcación del flujo, entradas y salida.

Los fuentes, tanto de esta CLIST como de la siguiente, se encuentran disponibles en el CD que acompaña a la revista, figurando a la derecha de cada línea la numeración a la que se hace mención en el presente artículo.

El lenguaje CLIST permite el manejo automatizado de los comandos del sistema MVS de IBM para *Grandes Sistemas*.

Comentario : /* ... */

los comentarios van encerrados entre los delimitadores indicados
/* comentario */

PROC n

Esta es la primera sentencia de una CLIST. Siempre lleva como argumento una cifra que especifica el número de parámetros que se pasan de entrada al procedimiento o programa de comandos.

Si n = 0 indica que no se pasan parámetros, pero, en el caso de que se pasen son admitidos como valores por defecto.

Los parámetros pueden pasarse por posición o por palabra clave. El siguiente ejemplo ilustra esta distinción:

```
PROC 2 Lib('jmpdes.ejemplo') Pgm('progra_1')  
PROC 2 'JMPDES.EJEMPLO','PROGRAM_1'
```

Asignación : SET

Esta sentencia sirve para asignar un valor a una variable. Así, en la línea 20 del ejemplo citado, se asigna a la variable TEST el valor TESTNO. Es de resaltar que no hace falta enmarcar entre comillas los valores alfabéticos.
comparar las sentencias de las líneas 21,46,57

En un procedimiento de comandos o CLIST, las variables se identifican por el & que precede a su nombre, siendo necesaria su especificación en todas aquellas sentencias que pueden dar lugar a confusión, no así en la sentencia SET ya que esta palabra clave debe ir seguida de un nombre de variable.

Bifurcación : IF

Esta sentencia permite asociar la ejecución de una o varias instrucciones al cumplimiento de una condición. Así, continuando con el ejemplo, la línea 22 permite activar unas opciones u otras para la depuración de la CLIST, en función del valor que contenga la variable TEST.
Cuando se desea condicionar varias sentencias a una condición, estas deben ir enmarcadas entre las palabras clave DO y END.

Depuración : CONTROL

Esta sentencia establece los parámetros que el sistema proporciona para el seguimiento de la ejecución a fin de permitir la depuración de errores.

```
SYMLIST muestra líneas de la CLIST  
CONLIST muestra los comandos que se ejecutan de TSO  
MSG muestra mensajes
```

END(xx) establece xx como la palabra que servirá de delimitador del DO. En la línea 25 se establece el valor "END" como delimitador, pues es el argumento que se pasa al parámetro END de la sentencia CONTROL.

Función cadena: &STR()

Esta función, identificada por ir precedida de & y seguida de (), convierte en cadena alfabética el argumento encerrado entre paréntesis. Así, en la línea 31 del ejemplo, lo que se está comparando es el valor de la variable &LIBRERIA con la cadena vacía, ya que no contiene argumentos.

Mensajes al usuario : WRITE

Esta sentencia escribe en la pantalla del usuario que está ejecutando el procedimiento todo lo que se especifique a continuación hasta el final de la línea. En el caso de que se especifiquen variables, se escribe el contenido, no el nombre de la misma.

Una variante de esta palabra clave es **WRITENR** donde se ha añadido NR (no return) a la palabra clave WRITE. Con esta sentencia se dice que escriba el mensaje como si se tratara de la sentencia WRITE, pero que no ejecute un retorno de carro. Esta opción suele usarse para enviar un mensaje al usuario para que este sepa lo que debe contestar (ver READ).

Entrada de datos : READ

Esta sentencia permite, en combinación con las anteriores, establecer un diálogo con el usuario, ya que detiene la ejecución del procedimiento e incluye lo escrito por el usuario en la variable que sirve de parámetro a esta palabra clave.

Desvío incondicional : GOTO

Esta sentencia desvía incondicionalmente el flujo de ejecución al punto etiquetado con el nombre que sirve de parámetro a esta sentencia. Este valor puede ser fijo o estar contenido dentro de una variable, como se ve al tratar la siguiente CLIST (JMP02).

Función &SYSDSN

Esta función del sistema valida la existencia de un fichero en el catálogo del sistema, devolviendo el valor OK cuando existe un fichero (particionado o secuencial) con el nombre especificado como argumento de la función. Para poder comprobar este extremo se recomienda cargar la variable &TEST con el valor TEST en la línea 20.

Invocar otra CLIST

La penúltima línea del ejemplo lo que hace es ejecutar un comando de usuario, es decir otro procedimiento, invocándole de forma explícita y pasando a este nuevo procedimiento unos parámetros, tal y como se comentó el pasado mes.

Fin de procedimiento: EXIT

Esta sentencia establece el fin de la CLIST, devolviendo el control al procedimiento desde el que se invocó a la misma.

2.- CLIST JMP02

La utilidad comentada en el punto anterior, simplemente sirve para iniciar un diálogo con el usuario que ejecute dicho procedimiento, pues sirve simplemente para pedir datos y validar los mismos.

La siguiente CLIST, podría definirse como una función ya que se asemeja su funcionamiento al de una caja negra que recibe unos datos de entrada (librería y posición) y concatena dicha librería a la DDNAME SYSPROC en la posición seleccionada, lo cual permitirá generalmente invocar un procedimiento de usuario usando el modo implícito, o bien saltarse algún control de seguridad por haber variado el orden de las librerías.

Básicamente la ejecución de esta CLIST supone el proceso secuencial de los bloques que esquemáticamente se representan en la figura 1 y que se pasan a comentar a continuación.

001-012 Definiciones generales:

004 : recoge los parámetros de entrada por posición, no por palabra clave.

006-011 : Muestran como se puede jugar con la variable **&TEST** para realizar distintos niveles de seguimiento para depuración. El autor es consciente de que se ha excedido en estas instrucciones, pero ha preferido mantenerlas simplemente por claridad didáctica, ya que los conceptos que se van a tratar aquí son mas complejos que los vistos anteriormente. Es de resaltar que el desarrollo de la misma se ha ampliado mas de lo debido a fin de ofrecer al lector mediante un ejemplo concreto, un mayor numero de sentencias, lo cual hace que este ejemplo sea mucho mas completo de lo que necesitaría el desarrollo de la utilidad propuesta,

0014-0025 Listar las librerías alocadas

016 : se desactivan las opciones LIST y CONLIST.

017 : carga la variable del sistema &SYSOUTTRAP con el valor 9999. Con lo cual se le dice al sistema que las siguientes 9999 líneas no las envíe al terminal, sino que las pasa a memoria.

018 : Ejecuta el comando de TSO : LISTALL con la opción STATUS para que liste no solo las librerías que tiene alocadas el usuario, sino para que liste las DDNAMES a las que están asignadas, tal y como se comentó el pasado mes, y se muestra en la figura 2.

La salida de este comando es desviada a memoria por la ejecución de la línea anterior. El comando no va precedido de la palabra clave TSO ya que cuando se esta ejecutando la CLIST, sus sentencias son interpretadas por TSO.

019 : restaura como valor de &SYSOUTTRAP el valor 0 para que los mensajes se dirijan de nuevo al terminal del usuario.

020 : Carga la variable de usuario &ULTIMO con el numero de líneas memorizadas, y que se encuentra contenido en la variable del sistema: &SYSOUTLINE.

0027-0043 Memorizar librerías listadas

Las sentencias que constituyen este bloque tienen por objetivo cargar una variable de usuario de tipo indexado o tabular (tabla) desde las variables del sistema en las que se encuentra la salida del comando ejecutado.

En este sentido, cabe resaltar que a diferencia de otros lenguajes, este tipo de variables no necesitan de una definición previa, sino que su nombre se forma concatenando un valor alfanumérico con un valor numérico a fin de obtener una serie de variables tales como ELEM001 ELEM002 ... ELEM099.

Pues bien, teniendo en cuenta lo dicho, las sentencias 28 a 35 constituyen una repetitiva (con GOTO incluido) para tratar cada una de las variables que formarán la tabla.

La variable &NOMBRE contiene "TABLA", que es la parte alfanumérica del nombre de la variable, mientras que la variable &CONTA constituye el índice de la tabla, con lo cual resulta que las variables a cargar son TABLA1, TABLA2, ... TABLA99, ...

Las líneas 30 y 31 muestran como cargando una variable con el valor de una etiqueta, se puede bifurcar desde un mismo GOTO n direcciones distintas.

La línea 33 es la clave de la asignación. No esta de más recordar que este lenguaje es un lenguaje interpretado, por lo que cuando se encuentra con &NOMBRE&CONTA lo traduce a TABLA1 (suponiendo que &CONTA valga 1); y, por otra parte, cuando se encuentra con &&SYSOutline&CONTA lo convierte a &&SYSOUTLINE1. En este caso, el doble ampersand (&&) que precede a la variable del sistema SYSOUTLINE especifica que se esta tratando una variable del sistema

El bucle de las líneas 36-42 solo se ejecuta cuando se esta en modo de TEST y permite listar la variable tabular (array) de usuario que se ha cargado en el bucle anterior. Esto no se puede hacer dentro del bucle anterior pues de hacerlo quedaría alterado el contenido de las variables del sistema &&SYSOUTLINE.

0043 Grabar en fichero librerías listadas

El siguiente bloque, innecesario desde el punto de vista de la utilidad, pero posiblemente útil para el lector, consiste en grabar el contenido de estas variables indexadas (o array de usuario) en un fichero o *dataset* físico. Lo primero que se debe hacer para realizar este cometido, es definir los atributos físicos de este fichero, cosa que se hace en las líneas 47-52.

043 : El nombre físico del fichero se forma concatenando la variable del sistema que contiene el identificador del usuario (&SYSUID) con el literal fijo .JMP.COMD (notar que después del primer cualificador hay dos puntos (..), ya que el primer punto '.' que sigue a SYSUID es el delimitador del nombre de la variable, y no pasa a formar parte del nombre del fichero).

047 : Examina si ya se encuentra en el catalogo un fichero con el nombre físico generado.

048 : Si ya existe, le *alloca*, es decir le incluye en la definición de su entorno asignándole el nombre lógico SALIDA, además con disposición OLD, es decir, para uso exclusivo, y borra el contenido anterior del mismo.

049-051 : Si no existe, incluye en su entorno el nombre lógico o DDNAME SALIDA, y la relaciona con el nombre físico contenido en la variable &FICHERO. El hecho de ir entre comillas esta variable, es para indicar al sistema, que el nombre es exacto, y que por tanto, no debe colocar el prefijo que tenga asignado el usuario en su perfil y comentado el pasado mes. El resto de parámetros son los que necesita el sistema operativo para crear y catalogar un fichero, tales como longitud de registro (LRECL), tipo de unidad sobre la que se almacena (discos 3380), tipo de formato del registro (fijo y bloqueado), espacio asignado (3 1) Tipo de unidad en la que esta expresada esta medida de espacio (TRACKS), Disposición en la que se encuentra el fichero al ejecutarse este comando (no existe, es nuevo : NEW), Disposición al completarse el paso bien (será incluido en el catalogo del sistema: CATALOG)

Las líneas 53-55 muestran como se puede escribir un literal en el fichero:

053 : Se abre como OUTPUT (salida) el fichero lógico SALIDA

054 : se carga el buffer del nombre lógico (&SALIDA) con el literal a grabar

055 : se escribe en el fichero el contenido del buffer cargado anteriormente.

0057-0070 Tratamiento rutina de errores

Una vez definido, *allocado* y abierto normalmente el fichero de salida, se pasa a comentar las siguientes líneas que componen la rutina de errores. Es importante resaltar que a esta rutina se cederá el control cuando la variable &LASTCC (Ultimo código de condición) que es del sistema, se cargue con un valor distinto de 0 al ejecutar cualquier instrucción.

Esta rutina esta activa desde el momento en el que el flujo de ejecución pase por ejecutar la sentencia ERROR , y hasta que dicho flujo ejecute la sentencia ERROR OFF, o se salga del procedimiento.

El hecho de que &LASTCC se cargue con un valor distinto de cero, no presupone que se haya producido un error, sino que se ha producido un estado de excepción que debe ser examinado, por ejemplo, se ha detectado el fin de fichero en lectura, lo cual devuelve un valor 400.

Las sentencias que integran esta rutina, en el caso de ser varias, se enmarcan por las palabras clave DO y END.

060 : Carga la variable de usuario &RC con el valor de &LASTCC ya que como se van a seguir ejecutando instrucciones, el valor de esta variable del sistema va a variar.

0071-0127 repetitiva para grabar línea

El siguiente bloque escribe en el fichero anterior las distintas librerías que se memorizaron previamente en la variable indexada. Este proceso le realiza basándose en las variables auxiliares &LINEA1 y &LINEA2, y manteniendo &CONTA como índice, el cual es inicializado a 0 en la línea 73.

Para su mejor comprensión, se recuerda que el comando LISTA STATUS escribe una línea por cada valor devuelto, sea librería o DDNAME, tal y como se muestra en la figura 2; y que lo primero que hace esta CLIST comentada (JMP02) es cargar una variable indexada con el valor de cada línea.

080 : Carga &LINEA1 con el valor de TABLAXX

081 : Carga &LINEA1, haciendo uso de la función &STR (String) con el valor resultante de concatenar el valor anterior y la variable &BLANCOS inicializada en la línea 74 con 30 blancos. Esta sentencia es necesaria ya que se pretende

ampliar la longitud del registro, y daría un error en el caso de no hacerlo así.

082 : Carga &LINEA1 con las 44 primeras posiciones de &LINEA1, lo cual se consigue haciendo uso de la función &SUBSTR (Substring). Esta función recibe como argumentos la posición del byte de inicio, la posición del byte final, y la cadena de la que extraer la información, que en este caso esta representada por el nombre de una variable. La subcadena extraída, es tratada como argumento de la función &STR para que en el caso de contener asteriscos, signos + o - , no sean considerados como operadores, sino como caracteres.

Una vez hecho esto, se examinan las 4 primeras posiciones de la variable &LINEA1 para ver si contienen los valores TERM o NULL. Estos son unos valores especiales, ya que se refieren al terminal, y solo devuelve una línea. En este caso, se carga la otra variable &LINEA2 con el contenido de las posiciones 9 a 23 de la cadena contenida en la variable &LINEA1; y bifurca el control a la etiqueta SEGUIR (123) para escribir en el fichero.

En el caso de no cumplirse la condición anterior, incrementa el índice, y carga en &LINEA2 el contenido del siguiente elemento de la tabla TABLAXx y examina las 10 primeras posiciones, ya que en esta posiciones de la segunda línea figura el nombre de la DDNAME o nombre lógico.

En el caso de que no contengan nada, carga:

- la variable de usuario &DDAUX con los caracteres 11 a 44 de &LINEA2, pues es en estas posiciones en las que figura la disposición de la librería (KEEP: Mantener/ SHR: Compartida/ OLD: Exclusiva) ;y

- la variable de usuario &DISPOSICION con la cadena que resulta de concatenar el contenido de la variable &DDNAME , una coma (,) ; y el valor de &DDAUX anterior, para que resulte el formato de la figura 3.

Y, en el caso de que esas primeras 10 posiciones no fueran blancos, es decir, cuando cambia la DDNAME, memoriza el nuevo valor en la variable de usuario &DDNAME para que sirva para las librerías siguientes, y carga en &DISPOSICION toda la línea leída, pues ajusta a la izquierda, y solo contiene la disposición, para configurar el registro de salida de la lista.

Se resalta el hecho de que el nombre de la DDNAME solo figura en la siguiente línea de la primera librería asociada a esta DDNAME, por eso se memoriza hasta que cambie.

107-110 : son líneas escritas para depuración.

111-118 : en el caso de que la DDNAME leída sea SYSPROC se carga otra variable indexada con los nombres de las distintas librerías asociadas a esta DDNAME tan especial.

119-127 : carga el buffer del fichero de OUTPUT : &SALIDA concatenando las variables &LINEA1 y &LINEA2 y escribe un único registro con el nombre de la librería, DDNAME (solo si cambia), y disposición, para que resulte un registro del formato mostrado en la figura 3.

0134-0172 repetitiva para tratar SYSPROC

Llegados a este punto ya tenemos todos los elementos que se necesitan para realizar de forma automática el objetivo propuesto para esta CLIST, a saber: *Incorporar la librería recibida como parámetro a la lista de librerías concatenadas a la DDNAME SYSPROC, y justamente en la posición que también se recibe como parámetro. Pero, si el parámetro de posición recibido fuera cero (0) debe eliminarse de dicha lista, la citada librería.*

Pues bien, una vez visto el bloque anterior, seguir el listado de la CLIST correspondiente a este bloque es fácil, y por eso el autor deja que sea el propio lector quien realice el seguimiento de las líneas 136-172.

174 : Al llegar al párrafo etiquetado con FIN se cuenta con una variable &LISTA_LIB que contiene la lista de librerías asociadas a SYSPROC; y con la variable &CONTA que contiene el numero de librerías incluidas en la variable anterior. Así, en el caso de que el numero de librerías relacionadas sea menor que el valor de la posición recibida como parámetro, se concatena el nombre de la librería leída a dicha lista, en caso contrario, significa que ya se ha incorporado en la posición que le correspondiera.

0188 : Alocar lista de librerías generada.

Disponiendo ya de la nueva lista, solo queda liberar del entorno del usuario todas las librerías asociadas a la DDNAME SYSPROC, cosa que se hace ejecutando en la línea 183 el comando de TSO : FREE. En este punto se desea resaltar, que

aunque SYSPROC no esté asociado a ninguna librería, este nombre lógico existe para el sistema, y no se ha borrado.

A continuación, se ejecuta el comando de TSO : ALLOC ya comentado en el artículo anterior, para asociar a SYSPROC la lista de librerías que se encuentra contenida en la variable &LISTA_LIB.

0190 : FIN

190 : EXIT , retorno al procedimiento que invocó esta CLIST.

3.- Consideraciones finales:

Con esto se llega al final de una utilidad, que además de ser muy útil, ha servido para comentar la mayor parte de las sentencias usadas por este lenguaje. No se ha tocado para nada los paneles de ISPF ya que son temas independientes y se pospone para el próximo mes la asociación entre ambos productos. El autor espera, no obstante, que lo visto sirva de estímulo a los lectores, y que estos se animen a crear sus propias utilidades pues con ello conseguirán una mayor eficacia en su gestión; y, al mismo tiempo, se pone a su disposición para resolver cuantas dudas les surjan, pudiendo dirigir sus consultas a la redacción de la revista, o bien directamente a <jmpecco@nova.es>.

```

----- ISPF/PDF PRIMARY OPTION MENU -----
OPTION ==> tso lista status
-----
0 ISPF PARMS - Specify terminal and user parameters      USERID -JMPDES
1 BROWSE     - Display source data or output listings   TIME    -10:43
2 EDIT       - Create or change source data            TERMINAL -3278
3 UTILITIES  - Perform utility functions                PF KEYS -12
4 FOREGROUND - Invoke language processors in foreground
-----
Enter END command to terminate ISPF.
-----
--DDNAME---DISP--
LIBDES01.SIST.ISPLLIB
ISPLLIB KEEP
LIBDES03.PALALIB.ISPLOAD
KEEP
LIBSYS05.ISPMLIB
ISPMLIB KEEP
ISP.SISPMENU
KEEP

```

Figura 2: Resultado obtenido al ejecutar el comando TSO LISTA STATUS

```

----- ISPF/PDF PRIMARY OPTION MENU -----
OPTION ==> tso lista status
-----
--DDNAME---DISP--
LIBDES01.SIST.ISPLLIB      ISPLLIB KEEP
LIBDES03.PALALIB.ISPLOAD  KEEP
LIBSYS05.ISPMLIB          ISPMLIB KEEP
ISP.SISPMENU              KEEP
ISP.SISFMLIB              KEEP
LIBDES03.PALALIB.ISPMLIB  KEEP
LIBDES01.SIST.ISPMLIB     KEEP

```

Figura 3: Resultado de concatenar de dos en dos el resultado obtenido en la figura 2 al ejecutar TSO LISTA STATUS.