

## CLIST (I)

### 0.- Entradilla

El lenguaje de CLIST es un lenguaje de programación más. Esta razón, junto con las de sencillez, potencia y desconocimiento entre los programadores, han hecho que el autor elija este tema para comenzar la sección dedicada a los **Grandes Sistemas**.

### 1.- Introducción

Para iniciar esta nueva etapa, el autor ha creído conveniente empezar tratando el tema CLIST porque, entre otras cosas, en un *Gran Sistema*, son muchos los programadores que asocian el termino CLIST con el departamento de sistemas, y ya va siendo hora de que desaparezca ese tópico, pues mientras en el mundo del PC, es el mismo programador el que genera sus ficheros .BAT que contienen los comandos necesarios para la ejecución de una aplicación, en el mundo de los **Grandes Sistemas** no ocurre lo mismo, ya que el programador espera que alguien del departamento de sistemas le codifique la CLIST correspondiente.

Una vez establecida esta premisa, cabe preguntarse ¿porqué cada desarrollador no se genera sus propias CLIST para simplificar esas tareas rutinarias que necesita reteclear todos los días? A priori, el autor desconoce exactamente la respuesta a esta pregunta, pero si puede asegurar que en parte puede ser debido a la barrera que existe entre los departamentos de desarrollo y de sistemas en un CPD(Centro Proceso de Datos), y muy posiblemente porque tradicionalmente este lenguaje ha sido usado por el departamento de Sistemas para montar sus procedimientos, y si se conocen las reglas de este lenguaje se podría seguir el proceso que emplea el departamento de sistemas para establecer los controles de seguridad, ya que es un lenguaje no compilado y, en consecuencia, pueden seguirse sus fuentes.

### 2.- Lenguaje

En principio debe decirse que el lenguaje de CLIST es un lenguaje de comandos (de ahí su nombre Command LIST) donde los comandos que se ejecutan son precisamente los propios comandos de TSO. Aquel lector acostumbrado al uso del PC entenderá perfectamente que el lenguaje usado en los ficheros .BAT equivale a ir ejecutando cada uno de los comandos del DOS desde el Prompt del sistema. Pues de igual modo, una CLIST es un conjunto de sentencias propias del lenguaje y de comandos elementales de TSO (Time Sharing Operation).

Así, se entiende por CLIST:

- Lenguaje de alto nivel, interprete de comandos propios y de TSO.
- Fichero secuencial que contiene el procedimiento, o conjunto de comandos a ejecutar.

Es probable que el lector de esta sección esté familiarizado con los JCLs (Job Control Language). Por eso el autor se permite aclarar que éste es un lenguaje que por su funcionalidad se parece al JCL, en base a la capacidad que tiene para hacer uso de los recursos del sistema, pero se diferencia del JCL en que para su ejecución no es necesario enviar ningún trabajo a las colas del JES (Job Entry System), ya que él puede realizar este cometido directamente, de forma interactiva, tratando la información que introduce el usuario en el sistema y devolviendo mensajes al mismo.

A modo de ejemplo, la primera parte de la figura 1 muestra una CLIST usada para imprimir un fichero haciendo uso del programa de utilidad del sistema IEBGENER, mientras que la segunda parte muestra un paso de un JCL que realiza el mismo cometido con el mismo programa.

Como se verá a lo largo de éste y siguientes artículos, el lenguaje de CLIST es un lenguaje sencillo, pero no por ello deja de ser potente. El único requerimiento que impone este lenguaje es que para poder ejecutar un procedimiento de clist es necesario estar en una sesión de TSO, cosa lógica por otra parte, pues los procedimientos hacen uso de todos los comandos de TSO. Así mismo, también puede invocarse un procedimiento de CLIST desde cualquier lenguaje de programación de aplicaciones, COBOL o Natural por ejemplo, siempre que se este ejecutando esa aplicación bajo TSO.

Una combinación muy interesante es la que integra este lenguaje con los paneles de ISPF para crear diálogos, tema este que también se tratará en próximos artículos.

Por otra parte, al ser un lenguaje intérprete, no necesita de compilación previa, por lo que una vez escrita la secuencia de comandos a ejecutar, esta puede ejecutarse directamente.

Por ultimo, también cabe destacar que este lenguaje de CLIST se puede usar para escribir aplicaciones, si bien las sentencias para la manipulación de datos, o de bifurcación no gozan de la flexibilidad que tienen en otros lenguajes orientados al manejo de datos.

### 3.- Librerías de CLISTS

Como ya se ha dicho en la introducción, una clist no es más que un fichero secuencial, y en consecuencia se almacena en el sistema como tal. No obstante, y con objeto de facilitar grandemente, entre otras, las tareas mecánicas del desarrollo, el sistema entiende distintos defectos para la ejecución de CLISTS.

El nombre del fichero o dataset que contiene la clist, ya sea un fichero PDS (Partitionated Data Set), o secuencial, puede tener hasta 44 caracteres, y debe seguir las reglas generales para la construcción de los nombres de ficheros en el entorno de MVS (Multiple Virtual Storage).

Se recomienda usar ficheros particionados(PDS) para almacenar CLISTS por las dos razones siguientes:

a) Normalmente las CLIST son ficheros pequeños, y en consecuencia, la gestión de su almacenamiento físico es gestionada mejor en una librería que en un fichero secuencial.

b) Permite la agrupación lógica de CLISTS.

Por su parte, el registro del fichero que contenga una CLIST, debe cumplir los siguientes requisitos:

- Si es de longitud fija: no puede superar los 255 bytes de longitud, reservando los últimos 8 bytes para el numero de secuencia.

- Si es de longitud variable: como tiene 4 bytes añadidos a la cabecera del registro para especificar la longitud del registro, éste puede tener una longitud de hasta 259 bytes, incluyendo los 4 de cabecera, y, la secuencia, que se especifica en los 8 bytes siguientes a los de longitud.

En general, se recomienda el uso del formato de longitud fija, no por nada en especial, sino porque admite tanto los ficheros numerados (NUMBER ON) como los no numerados (UNNUM).

### 4.- Cómo invocar una CLIST

Una CLIST puede invocarse por una de las dos forma siguientes:

**a) De forma explícita:** Invocando al procedimiento mediante la palabra clave EXEC, tal y como se muestra en los siguientes ejemplos:

```
COMMAND ==>> tso ex 'JMPDES.clist(lnatddm)', 'test'
```

En este caso, se invoca una clist pasando como parámetro la cadena TEST.

Los siguientes ejemplos muestran cómo el propio TSO nos indica cual es el defecto que entiende cuando se invoca una clist erróneamente.

```
COMMAND ==>> tso ex (lnatxxx)
```

```
IKJ56520I MEMBER LNATXXX NOT IN DATASET JMPDES.CLIST
***
```

En este caso, el sistema busca el dataset 'JMPDES.CLIST(LNATXXX)' y como no le encuentra, devuelve el mensaje que se acompaña.

```
COMMAND ==>> tso ex clist.int(lnatddm)
```

```
IKJ56228I DATA SET JMPDES.CLIST.INT.CLIST NOT IN CATALOG OR CATALOG CAN
NOT BE ACCESSED
***
```

En este ejemplo, al no haber encerrado entre comillas el nombre de la CLIST, y, si en el perfil del usuario figura <PREFIX(jmpdes)>, se espera que el sis-

tema anteponga el prefijo del usuario, tal y como hace con cualquier nombre de fichero, pero el resultado es otro, ya que por defecto el sistema no solo antepone el prefijo, sino que también pone como sufijo de la librería que es invocada con EXEC la cadena 'CLIST', con independencia de que se llame a un procedimiento que sea miembro de un Particionado, o que se encuentre en un fichero secuencial. El siguiente ejemplo ilustra este caso:

```
COMMAND ==> tso ex JMP
COMMAND ==> tso ex 'JMPDES.JMP.CLIST'
```

En ambos casos se está invocando al mismo procedimiento, siempre en el supuesto de que el usuario tenga definido JMPDES como PREFIX en su perfil.

Para saber como se encuentra definido, ejecutar desde línea de comando: < TSO PROFILE > y, si el parámetro que devuelve es NOPREFIX, ejecutar < TSO PROFILE PREFIX(jmpdes) > y a partir de ese momento, a cualquier procedimiento invocado sin comillas el sistema le antepondrá el prefijo especificado.

#### **b) De forma implícita**

La otra forma de invocar una CLIST, consiste en invocar procedimientos que se encuentran en las librerías alocadas para tal efecto.

Para poder entender este concepto, el lector debe asimilar previamente que el **entorno** que tiene asignado cada uno de los programadores de un gran sistema, se le define expresamente a cada uno cuando ejecuta el procedimiento de LOGON o conexión al TSO. Este entorno se encuentra compuesto por una serie de ficheros lógicos, no físicos, que reciben el nombre de **Ddname** (Data Definition Name). Cada Ddname tiene asociadas una serie de ficheros o librerías físicas, en las cuales el sistema buscara el elemento que necesite en cada momento. Esta búsqueda la realizara solo en la librerías asociadas a la Ddname correspondiente al tipo tratado. Así, los paneles de ISPF les buscara en las librerías asociadas a la Ddname ISPPLIB, los mensajes en las librerías asociadas a la Ddname ISPMLIB, y los procedimientos de comandos o CLIST se encuentran asociados a la Ddname SYSPROC.

Una DDNAME es el nombre dado a un fichero lógico, no físico, pero que tiene un significado especial.

El tenerlo de esta forma establecido permite modificar la lista de librerías asociadas a un nombre lógico o DDNAME sin necesidad de redefinir nuevos nombres lógicos de ficheros. Esta modificación se realiza mediante el comando ALLOC que se vera en el siguiente apartado.

Pues bien, TSO para la búsqueda del comando a ejecutar, examina todas las librerías de CLIST alocadas en el sistema, *en el orden inverso* al de su alocación. Entre otras cosas, este punto este punto tiene su importancia, ya que puede permitir que se anulen los efectos de una clist de uso general, creando simplemente otra clist con el mismo nombre que la de uso general, pero incluyéndola en las ultimas librerías de la lista asociada a SYSPROC, para que de esta forma, cuando se invoque dicho procedimiento encuentre, y en consecuencia se ejecute, antes la particular que la general.

Cuando se desee ejecutar una clist siguiendo el orden directo, se debe anteponer al comando el símbolo %

El siguiente ejemplo muestra la forma de invocar un procedimiento de clist usando la forma implícita, y además buscando solo en la librería SYSPROC.

```
COMMAND ==> tso %NATDESA
```

Este procedimiento establece el entorno de NATURAL-DESARROLLO y nos introduce en el.

#### **4.- librerías ALLOCADAS**

Antes de nada, el autor quiere pedir disculpas por el palabro utilizado(lo cual también es otro palabro), pero al ser un término tan utilizado en estas instalaciones ha llegado a incorporarse a su vocabulario.

Realmente el termino proviene del comando de TSO ALLOC abreviatura de ALLOCATE, el cual significa asignar al entorno. No significa reserva de espacio, con el que se le asocia normalmente, lo único que hace es incorporar *el nombre de la librería* que se especifique a la Ddname que se indique. Comparando este co-

mando con el DOS, se podría decir que alocar una librería significa incluir dicha librería (o directorio del DOS) en el PATH del sistema, para que pueda localizar en tiempo de ejecución cualquier fichero sin necesidad de que se encuentre en el mismo directorio.

Para conocer las librerías que se encuentran alocadas, es decir, definidas o asignadas en el entorno de cada usuario, se puede ejecutar el comando de TSO: LISTALL con la opción **STATUS**.

la figura 2 muestra el resultado de dicha ejecución, aunque, como puede observarse, el resultado necesita dos líneas por librería. Concatenando ambas líneas, el resultado sería el de la figura 3, la cual muestra otra forma de ver la misma lista de librerías alocadas para cada una de las Ddnames que constituyen el entorno del usuario, así como la disposición que tiene cada librería física, pero de una forma algo mas inteligible.

Así, la Ddname ISPLLIB, librería de módulos ejecutables de ISPF (ISP-L-LIB) tiene dos librerías físicas asignadas, mientras que los mensajes que nos devuelva el sistema se pueden encontrar en cualquiera de las 5 librerías asociadas a la Ddname ISP-M-LIB.

## 5.- Utilidad

Por ultimo, y para cerrar esta primera entrega, se acompaña el listado de la primera parte de una utilidad que se ha dividido en dos partes para mostrar de forma practica las distintas formas de invocar una CLIST, incluso desde otra CLIST.

En definitiva, la utilidad que se acompaña, y cuyos fuentes se encuentran en el CD que acompaña a la revista, tiene por nombre JMP01, y al ser invocada de forma explicita, permite recoger y validar dos parámetros, compuestos por nombre de librería y posición. Una vez validados estos parámetros, son pasados también como parámetros a la CLIST JMP02, la cual se encarga de leer del entorno las librerías asociadas a la Ddname SYSPROC. Y, a continuación, incluye en la posición elegida la librería recibida, o la elimina del entorno si la posición elegida es la 0.

En el próximo artículo se tratará con mas detalle la sintaxis de las sentencias incluidas en ambos procedimientos, pero se ha optado por incluir ambas CLIST en esta primera entrega a fin de el lector pueda ejecutar la utilidad íntegramente y hacer sus propios pinitos, ya que únicamente debe copiar dichos fuentes en el HOST mediante un simple file-transfer para tener esta utilidad ejecutable en su entorno.

```

/* ===== imprimir fichero con CLIST ===== */
/*
ALLOC FILE(SYSUT1)    DATASET('JMPDES.TEMPORAL') SHR REUSE
ALLOC FILE(SYSUT2)    SYSOUT(X) DEST(RMT101) REUSE
ALLOC FILE(SYSPRINT)  DUMMY                                REUSE
ALLOC FILE(SYSIN)     DUMMY                                REUSE
CALL 'SYS1.LINKLIB(IEBGENER)'
FREE FILE(SYSUT1 SYSUT2 SYSPRINT SYSIN)

```

```

/**
----- imprimir fichero con JCL -----
//PASO00 EXEC PGM=IEBGENER
//STEPLIB DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSUT1 DD DSN=JMPDES.TEMPORAL,DISP=SHR
//SYSUT2 DD SYSOUT=X,DEST=RMT101
//SYSPRNT DD DUMMY
//SYSIN DD DUMMY

```

Figura 1

```

----- ISPF/PDF PRIMARY OPTION MENU -----
OPTION ==> tso lista status

                                USERID  -JMPDES
0 ISPF PARMS  - Specify terminal and user parameters  TIME      -10:43
1 BROWSE      - Display source data or output listings  TERMINAL  -3278
2 EDIT        - Create or change source data          PF KEYS   -12
3 UTILITIES   - Perform utility functions
4 FOREGROUND  - Invoke language processors in foreground

Enter END command to terminate ISPF.

--DDNAME---DISP--
LIBDES01.SIST.ISPLLIB          ISPLLIB  KEEP
LIBDES03.PALALIB.ISPLOAD      ISPLLIB  KEEP
LIBSYS05.ISPMLIB              ISPMLIB  KEEP
ISP.SISPMENU                   KEEP

```

Figura 2

```

----- ISPF/PDF PRIMARY OPTION MENU -----
OPTION ==> tso lista status

--DDNAME---DISP--
LIBDES01.SIST.ISPLLIB          ISPLLIB  KEEP
LIBDES03.PALALIB.ISPLOAD      ISPLLIB  KEEP
LIBSYS05.ISPMLIB              ISPMLIB  KEEP
ISP.SISPMENU                   KEEP
ISP.SISFMLIB                   KEEP
LIBDES03.PALALIB.ISPMLIB      ISPLLIB  KEEP
LIBDES01.SIST.ISPMLIB         ISPLLIB  KEEP

```

Figura 3

