INTRODUCCION AL SISTEMA VAX-11/780

D C L

E D I T O R E D T

R U N O F F

Eduardo S. Jallath Coria
Alejandro Jiménez García
Sócrates A. Muñiz Zafra
Humberto Sánchez Sandoval

1 9 8 3

Centro de Cálculo de la Facultad de Ingeniería

El lenguaje de comandos DCL tiene una construcción de tipo arborescente, donde el primer nivel corresponde a los diferentes comandos de DCL y, en los subsiguientes niveles, se encuentran las diferentes opciones de cada comando.

En la figura 1 se muestra un ejemplo de éste.

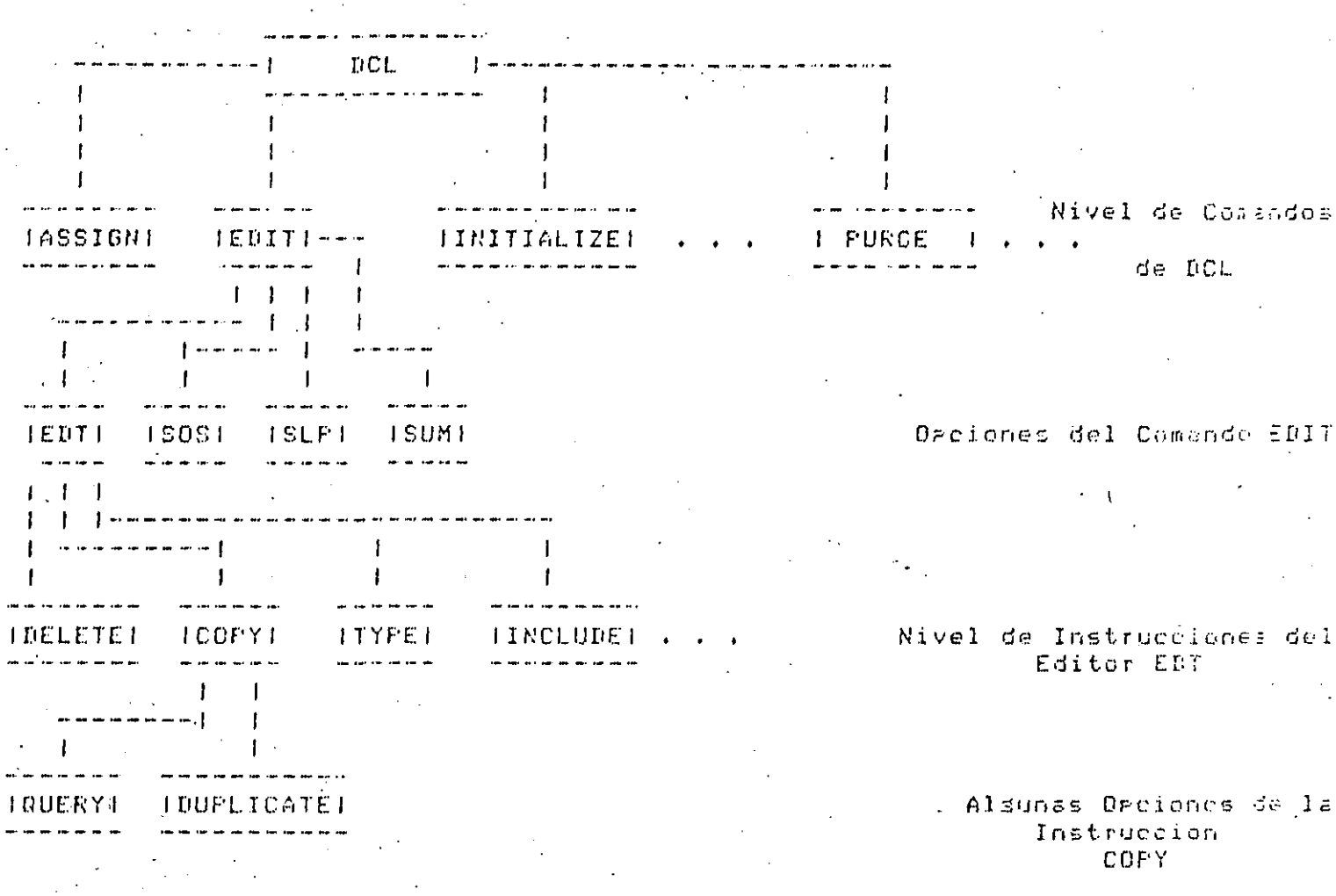
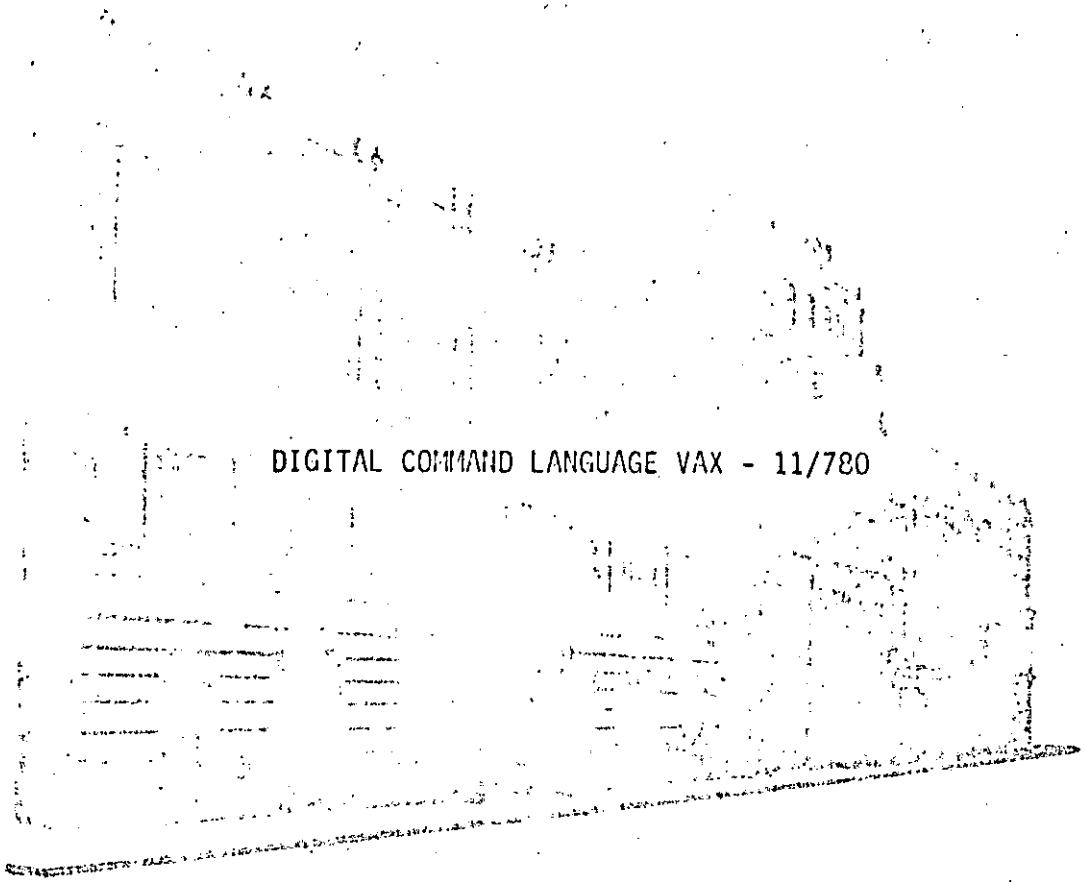


Fig. 1.

A continuación se muestran los comandos básicos utilizados para poder tener acceso al sistema, así como los comandos necesarios para la edición de programas.

CURSOS: "INTRODUCCION AL SISTEMA VAX-II/780" Y "OPERACION Y ADMINISTRACION DEL SISTEMA VAX/VMS"

DEL 29 DE ABRIL AL 13 DE MAYO
MEXICO. D.F.



PROFESORES:

ING. EDUARDO S. JALLATH CORIA.
ING. ALEJANDRO JIMENEZ GARCIA.
ING. SOCRATES A. MUNIZ ZAFRA.
ING. HUMBERTO SANCHEZ SANDOVAL.

MAYO DE 1985.

CAPITULO 1

DIGITAL COMMAND LANGUAGE VAX - 11/780

1.1 INTRODUCCION

El lenguaje de Comandos de VAX, llamado DCL (Digital Command Language), es el primer lenguaje de alto nivel diseñado para comunicación interactiva, y por BATCH con el mismo conjunto de instrucciones.

Esto representa un gran adelanto, dado que tradicionalmente se utiliza un lenguaje de comandos para comunicación interactiva y otro para comunicación por BATCH.

En el caso de DCL se usa un solo lenguaje para ambos tipos de comunicación, lo cual facilita su aprendizaje, ésto aunado a su poderío nos permite comunicación amplia y extensa con el sistema VAX-11/780.

El lenguaje DCL proveé al usuario VAX con un extenso conjunto de instrucciones para:

Desarrollo interactivo de programas.

Ejecución y control de programas por BATCH.

Ejecución y control de programas en tiempo real.

Manipulación de dispositivos de entrada y salida.

Manejo de archivos de información.

Este instructivo pretende ser un texto introductorio al sistema VAX-11/780, razón por la cual, y para facilitar ciertas descripciones es posible que algunos comandos sean presentados de una manera poco riguroso y sin todos las opciones del mismo. Es por ello que no debe ser tomado como un curso formal, sino como un manual de acceso rápido al sistema.

.2 FORMATO DE LOS COMANDOS

Los comandos son palabras del idioma inglés que describen la acción que se desea ejecutar.

Estos comandos pueden, opcionalmente, contener calificadores u opciones, que modifican la acción del comando.

Ejemplo:

$ PRINT EJEMPLO.DAT

Este comando manda a la cola de impresión un listado del archivo EJEMPLO.DAT

$ PRINT/COPIES=2 EJEMPLO.DAT

La acción de este calificador es mandar 2 copias del archivo especificado a la cola de impresión.

Todos los calificadores van separados por una diagonal (/).

Ejemplo:

$ PRINT/COPIES=20/AFTER=23:00:00 EJEMPLO.DAT

Manda 20 copias del archivo EJEMPLO.DAT a la cola de impresión, pero éstas no se imprimirán hasta después de las 23 horas.

Cuando alguno de los parámetros del comando no es especificado, el sistema lo pide. De este modo, nos ayuda llevándonos de la mano.

Ejemplo:

$ PRINT

$_File: EJEMPLO.DAT

Dado que el comando PRINT requiere de un parámetro, y éste no fue proporcionado por el usuario, el sistema pide el parámetro por medio del prompt FILE.

La forma general de un comando de DCL es

$ Comando/calif1/calif2. . . parámetro calif3/calif4. . .

Calificador 3 y calificador 4 son calificadores de los parámetros del comando, mientras que calificador 1 y calificador 2 son calificadores del comando.

1.3 LA TERMINAL VT100

La terminal es el medio a través del cual nos comunicamos con la computadora, es por ello que conviene conocer algunas de sus características.

A pesar de que parece una máquina de escribir común y corriente, tiene 3 diferencias esenciales:

- Interpretación de caracteres mayúsculos y minúsculos
- El Buffer de Almacenamiento
- Teclas de función especial.

a) Cuando se teclean comandos en minúsculas, el intérprete de DCL se encarga de cambiarlas a mayúsculas y mandar así el comando a la computadora.

b) Después de mandar un comando, y mientras el intérprete de DCL lo ejecuta, el teclado no se bloquea, ésto es, el usuario puede seguir tecleando aunque ésto no aparezca en la terminal, ya que contiene un buffer de almacenamiento para este fin. Cuando el comando anterior ha terminado de ejecutarse, la terminal desplegará completo el nuevo comando que hemos estado tecleando hasta ese momento. En caso de haber mandado varios comandos, el sistema primero los despliega y luego los ejecuta.

c) Teclas de función especial.

En la tabla 1 se muestra un resúmen de las funciones que realizan algunas de las teclas de la terminal.

TABLA 1

i) RETURN (Carriage Return)

Transmite la línea actual hacia el sistema para su procesamiento.

(En algunas terminales, la tecla RETURN está etiquetada como CR).

Antes de una sesión de terminal, inicia la secuencia del Losin.

ii) Teclas de Control.

Definen las funciones a ejecutar cuando la tecla CTRL y otra tecla sean presionadas simultáneamente. Todas las secuencias de tecleo CTRL/x se ven en la pantalla como ^x.

iii) CTRL/C y CTRL/Y.

Si se dan durante un comando de entrada, cancela el procesamiento del comando.

Antes de una sesión de terminal, inicia la secuencia del Login.

Interrumpe el comando o la ejecución de un programa y regresa el control al intérprete de comandos.

iv) CTRL/I

Avanza el cursor a la siguiente posición del tabulador.

v) CTRL/K

Avanza la línea actual a la siguiente marca del tabulador.

vi) CTRL/L

Form Feed

vii) CTRL/O

Alternativamente, suprime y continúa mostrando la salida a la terminal.

viii) CTRL/Q

Restaura la salida de la terminal que fué suspendida por CTRL/S.

ix) CTRL/R

Reimprime la línea actual y pone el cursor al final de la línea.

x) CTRL/S

Suspende la salida a la terminal hasta que se presione CTRL/Q.

xi) CTRL/U

Suprime la línea actual.

xii) CTRL/X

Descarta la línea actual y borra datos en el buffer de almacenamiento.

xiii) CTRL/Y

Véase CTRL/C.

xiv) CTRL/Z

Señala el fin de archivo (EOF) para los datos metidos por terminal.

xv) CTRL/W

Actualiza la pantalla con la última página del archivo con que se está trabajando.

Para poder observar las características físicas de una terminal, será necesario dar el comando:

$ SHOW TERMINAL

Para poder modificar las caracteristicas de una terminal es necesario dar el comando:

$ SET TERMINAL

Mas adelante se verán en detalle estos dos comandos.

1.4 COMO ENTRAR AL SISTEMA

Para lograr la atención del sistema, basta con oprimir la tecla RETURN, o bien, simultáneamente CTRL/Y.

El sistema responde:

Que tengas una agradable sesion

Y pide el USERNAME.

Una vez que este último ha sido tecleado, el sistema pide el PASSWORD.

Se realiza el proceso de verificación de la clave del usuario y si ésta es válida, el sistema responde:

Bienvenidos al Sistema VAX/VMS V3.1 (C e c a f i)

7-OCT-1983 12:11:33.55
BUENAS TARDES

$

Siendo el simbolo ($) el que nos indica que estamos en el nivel de comandos de DCL, con lo cual podemos empezar a comunicarnos con el sistema.

Ejemplo:

```
-----------
|   RET   |
-----------

                    -----------
USERNAME: OSITO     |   RET   |
                    -----------

PASSWORD: PANDA          NOTA: El password no se desplega al ser
                               tecleado.


          Bienvenidos al Sistema VAX/VMS V3.1 ( C e c a f i )


    7-OCT-1983 12:11:33.55
    BUENAS TARDES

    $
```

1.5 COMO SALIR DEL SISTEMA

Una vez terminado nuestro trabajo con la VAX, basta dar el comando

```
LOGOUT/BRIEF
```

o bien,

```
LOGOUT/FULL
```

Con lo que terminará nuestra sesión de terminal.

1.6 COMANDO HELP

La VAX contiene una extensa biblioteca de ayuda al usuario, la cual es desplegada con el comando HELP.

En caso de no recordar la sintaxis específica de un comando basta teclear:

$ HELP <comando>

y con ello el sistema nos indicará las características y requerimientos del mismo.

Ejemplo:

$ HELP

Despliega los comandos existentes en DCL

$ HELP PRINT

Despliega las funciones y requerimientos del comando PRINT

$ HELP PRINT/COPIES

Despliega las funciones del comando PRINT afectado por el calificador COPIES.

Debe hacerse notar, que el comando HELP es un comando de AYUDA y no de ENSEMANZA, por lo que el usuario no deberá esperar aprender DCL a través del comando HELP.

1.7 COMANDOS DE DCL

Se presenta a continuación un resúmen de los comandos de DCL más utilizados, explicando su función y presentando las opciones más importantes de cada comando.

..7.1 COMANDO @

Ejecuta un procedimiento de comandos previamente almacenado en disco.

Formato:

$ @ ARCHIVO.COM

ARCHIVO.COM es el nombre del procedimiento de comandos que se desea ejecutar. Este archivo se asume de tipo .COM por default.

Ejemplo:

$ @ COMANDOS.COM

Opciones:

/OUTPUT = ARCHIVO2.LIS

La salida generada por la ejecución del procedimiento de comandos es mandada al ARCHIVO2.LIS especificado.

Ejemplo:

$ @COMANDOS.COM/OUTPUT = SALIDA.LIS

Durante la ejecución del procedimiento, la terminal realizará una a una las instrucciones contenidas en el mismo. Si se desea seguir paso a paso la ejecución de cada instrucción, antes de ejecutar el procedimiento, será necesario dar el comando:

$ SET VERIFY

1.7.2 COMANDO ASSIGN

Hace una asignación de un nombre lógico con un dispositivo físico, un chivo u otro nombre lógico.

Formato:

$ ASSIGN nombre-equivalente nombre-lógico

Nombre-equivalente: Especifica el nombre del dispositivo o archivo al que se le va a asignar un nombre lógico.

Nombre-lógico: Especifica el nombre lógico (1 a 63 caracteres) que se va a asociar con el dispositivo o archivo.

Ejemplo:

$ ASSIGN TTG3: SYS$PRINT

$ PRINT ARCHIVO

El nombre lógico del sistema SYS$PRINT (que es el área donde se ponen las impresiones) se asigna al dispositivo TTG3; al mandar la instrucción PRINT (la cual, por default, mandaría la impresión al área SYS$PRINT) provocará que la impresión salga en el dispositivo TTG3, el cual puede ser una impresora remota.

1.7.3 COMANDO BASIC

Se invoca al compilador BASIC para compilar un programa escrito en lenguaje BASIC.

Formato:

$ BASIC ARCHIVO.BAS

ARCHIVO.BAS es el programa fuente que será compilado.

Ejemplo:

$ BASIC PROG.BAS

Compila el programa fuente PROG.BAS y genera el archivo objeto PROG.OBJ

Ejemplo:

```
$ BASIC

VAX-11 Basic

Ready
```

Si se teclea BASIC sin especificar el nombre de archivo se inicia una comunicación interactiva con la computadora, funcionando BASIC como intérprete.

1.7.4 COMANDO COBOL

Invoca al compilador COBOL para compilar un programa escrito en lenguaje COBOL.

Formato:

```
$ COBOL ARCHIVO.COB
```

ARCHIVO.COB es el programa fuente que será compilado.

Ejemplo:

```
$ COBOL PROG.COB
```

Compila el programa fuente PROG.COB y genera el archivo objeto PROG.OBJ

1.7.5 COMANDO COPY

Utilizado para crear una copia de un archivo ya creado, dentro del mismo subdirectorio, de un subdirectorio a otro, de un dispositivo a otro, de una clave a otra, etc.

Formato:

$ COPY ARCH1 ARCH2

Ejemplo:

$ COPY

FROM: ARCH.PAS

TO: ARCHIVO.COB

Copia la información contenida en ARCH.PAS a un nuevo archivo ARCHIVO.COB.

1.7.6 COMANDO CREATE/DIRECTORY

Comando utilizado para la creacion de subdirectorios. Un subdirectorio aparece como un archivo al utilizar el comando DIRECTORY.

Formato:

$ CREATE/DIRECTORY [nombre subdirectorio]

Ejemplo:

$ CREATE/DIR [EJEMPLO.TAREAS]

Crea un subdirectorio en la clave EJEMPLO, cuyo nombre es TAREAS. Para posicionarnos en este nodo se utiliza el comando SET DEFAULT.

$ CREATE/DIR [.OTROS]

Crea un subdirectorio en el nodo (subdirectorio) en el que se este trabajando llamado OTROS.

1.7.7 COMANDO DEASSIGN

Cancela las asignaciones de nombres lógicos, hechas previamente con un comando ASSIGN.

Formato:

$ DEASSIGN nombre-lógico

Ejemplo:

$ ASSIGN RES.LIS SYS$OUTPUT

$ DEASSIGN SYS$OUTPUT

El comando ASSIGN asigna el nombre lógico SYS$OUTPUT al archivo temporal RES.LIS.

El comando DEASSIGN suprime esta asignación.

1.7.8 COMANDO DELETE

Borra uno o mas archivos almacenados en disco.

Formato:

$ DELETE lista-de-archivos

Ejemplo:

$ DELETE ARCH1.PAS;2, ARCH.BAS;*

Borra la 2a. versión del archivo ARCH1.PAS y además todas las versiones de ARCH.BAS

Siempre será necesario especificar el número de la versión.

Qué haría DELETE *.*;* ?

Cuidado! Borra todo.

Opciones:

/CONFIRM

Esta opción hace que el sistema pregunte antes de borrar un archivo.

Ejemplo:

$ DELETE/CONFIRM *.*;*

1.7.9 COMANDO DELETE/ENTRY

Sirve para descontinuar un JOB o un listado de impresión que esté en alguna cola de BATCH o cola de impresión.

Formato:

$ DELETE/ENTRY = (Lista-de-números-de-JOB's) nombre-de-cola

Número de JOB: Es el número asignado al JOB en la cola.

Número de cola: Cola de donde se van a descontinuar los JOB's.

Ejemplo:

```
$ PRINT/HOLD ARCH.TXT

Job 110 entered on queue SYS$PRINT

.

.

.

$ DELETE/ENTRY = 110 SYS$PRINT
```

La instrucción PRINT coloca en la cola SYS$PRINT el archivo ARCH.TXT con el número 110 y lo mantiene en ella (opción Hold), sin imprimirse hasta que se le indique posteriormente.

La instrucción DELETE/ENTRY borra de la cola SYS$PRINT el JOB 110, el cual no se imprimirá.

1.7.10 COMANDO DIRECTORY

Despliega una lista de los archivos almacenados en la clave del usuario.

Formato:

```
$ DIRECTORY [Lista de archivos]
```

Ejemplo:

```
$ DIRECTORY
```

Despliega todos los archivos con todas sus versiones

$ DIRECTORY *.PAS

Despliega todos los archivos de tipo .PAS

1.7.11 COMANDO EDIT

Invoca al Editor de Pantalla.

Formato:

EDIT <archivo>

<archivo> es el nombre del programa o archivo que deseamos editar.

Ejemplos:

$ EDIT ARCHIVO.FOR

Opciones:

/EDT Edita un archivo con el editor EDT

/SOS Edita un archivo con el editor SOS

/SUM Edita un archivo con el editor SUM

/SLP Edita un archivo con el editor SLP

Si no se especifica la opción, toma el default EDT.

Mas adelante se hace una descripción detallada de los comandos del editor EDT.

1.7.12 COMANDO FORTRAN

Este comando invoca al compilador FORTRAN para compilar un programa escrito en lenguaje FORTRAN.

Formato:

$ FORTRAN ARCHIVO.FOR

ARCHIVO.FOR es el nombre del archivo que se desea compilar.

Ejemplo:

$ FORTRAN PROG.FOR

1.7.13 COMANDO LINK

Este comando invoca al ligador del sistema, el cual servirá para unir el programa objeto con todas las demás rutinas externas del sistema.

Formato:

$ LINK ARCHIVO.OBJ,RUTINA.OBJ,OTRA.OBJ

ARCHIVO.OBJ es el programa objeto el cual va hacer ligado con las rutinas RUTINA.OBJ y OTRA.OBJ. Las dos rutinas se compilarón aparte.

Si todas las rutinas estan dentro del ARCHIVO.OBJ, sólo se ligan a éste.

Ejemplo:

$ LINK ARCHIVO.OBJ

1.7.14 COMANDO PASCAL

Este comando invoca al compilador Pascal para compilar un programa escrito en lenguaje Pascal.

Formato:

$ PASCAL ARCHIVO.PAS

ARCHIVO.PAS es el nombre del programa que se desea compilar.

Ejemplo:

$ PASCAL PROG.PAS

1.7.15 COMANDO PRINT

Este comando sirve para meter en la cola de impresión uno o mas archivos para imprimirse.

Formato:

$ PRINT ARCH1, ARCH2

Ejemplo:

$ PRINT ARCHIVO.PAS, ARCH1.COB

Opciones:

/AFTER = hora.

Con esta opción el listado se imprimirá después de la hora especificada.

/QUEUE = nombre-de-cola.

Imprime el archivo en la cola de impresión especificada; cada impresora o terminal-impresora tiene asociado un nombre de cola.

Ejemplo:

$ PRINT/QUEUE = LPB0:/AFTER = 20:23:36 ARCH.LIS

Imprime el archivo ARCH.LIS, en el dispositivo llamado LPB0:, después de las 20 horas 23 minutos y 36 segundos.

/DELETE

Borra el archivo de impresión cuando termine de imprimir.

1.7.16 COMANDO PURGE

Borra todas las versiones existentes de un archivo, excepto la más ciente (la de número de versión mayor).

Formato:

$ PURGE [archivos]

Ejemplo:

$ PURGE

Borra todas las versiones de todos los programas, excepto la más reciente.

$ PURGE *.COM

borra todas las versiones de los archivos de tipo .COM, excepto la más reciente.

Opciones:

/KEEP=n

Hace el PURGE de las versiones, dejando en disco exclusivamente el número especificado de las mismas.

1.7.17 COMANDO RENAME

Sirve para cambiar de nombre, tipo, versión etc., un archivo o un directorio.

Formato:

$ RENAME ARCH1 ARCH2

Ejemplo:

$ RENAME

FROM: ARCH.PAS

TO: ARCHIVO.COB

Cambia el nombre del archivo ARCH.PAS a ARCHIVO.COB

1.7.18 COMANDO REQUEST

Este comando es usado para mandar un mensaje al operador del sistema.

Formato:

$ REQUEST "mensaje"

Ejemplo:

$ REQUEST "HOLA SR. OPERADOR"

1.7.19 COMANDO RUN

Sirve para ejecutar un programa de tipo .EXE (es decir ya compilado y ligado).

Formato:

$ RUN ARCHIVO.EXE

Ejemplo:

$ RUN CORRE.EXE

$ RUN PROG.EXE

1.7.20 COMANDO STOP/ABORT

Aborta un JOB que se está imprimiendo.

Formato:

$ STOP/ABORT identificación-de-impresora

Ejemplo:

$ STOP/ABORT LPA0:

Descontinúa la impresión que esté saliendo por la impresora LPA0:

1.7.21 COMANDO STOP/ENTRY

Descontinua de la cola de BATCH el proceso que está corriendo.

Formato:

$ STOP/ENTRY = Número-de-Job nombre-de-cola

Ejemplo:

$ STOP/ENTRY = 230 SYS$BATCH

Descontinua de la cola SYS$BATCH al Job número 230.

1.7.22 COMANDO SUBMIT

Es usado para meter en cola de procesos BATCH uno o más procedimientos de comandos.

Formato:

$ SUBMIT ARCHIVO.COM

Ejemplo:

$ SUBMIT PROGRAMA.COM

Mete el archivo de comandos PROGRAMA.COM a la cola SYS$BATCH que es la cola de BATCH que existe por default.

Opciones:

/QUEUE=nombre-de-cola

Ejemplo:

$ SUBMIT/QUEUE=1 EJEMPLO.COM

Mete a la cola 1 el procedimiento de comandos EJEMPLO.COM

.7.23 COMANDO TYPE

Despliega uno o mas archivos en la terminal.

Formato:

$ TYPE archivo

Ejemplo:

$ TYPE MIARCHIVO.PAS

Listará el programa MIARCHIVO.PAS en la terminal.

CURSOS: "INTRODUCCION AL SISTEMA VAX - 11/780" Y
"OPERACION Y ADMINISTRACION DEL SISTEMA VAX /VMS"
DEL 29 DE ABRIL AL 13 DE MAYO.
MEXICO, D.F.

SHOW Y SET

PROFESORES:

ING. EDUARDO S. JALLATH CORIA.
ING. ALEJANDRO JIMENEZ GARCIA.
ING. SOCRATES A. MUÑIZ ZAFRA.
ING. HUMBERTO SANCHEZ SANDOVAL.

MAYO DE 1985.

CAPITULO 2

SHOW Y SET

2.1 COMANDO SHOW TIME

Despliega la fecha y hora actual.

Ejemplo:

```
$ SHOW TIME
```

2.2 COMANDO SHOW DEFAULT

Despliega el dispositivo y el directorio en el cual se encuentra la sesión. Tanto el dispositivo como el directorio pueden ser cambiados con el comando SET DEFAULT.

Ejemplo:

```
$ SHOW DEFAULT
```

2.3 COMANDO SHOW DEVICES NOMBRE-DE-DISPOSITIVO

Despliega el estado de los dispositivos que se encuentran conectados a sistema.

Ejemplos:

$ SHOW DEVICES

Despliega todos los dispositivos conectados al sistema.

$ SHOW DEVICES LPA0:

Despliega el estado de la impresora LPA0:

2.4 COMANDO SHOW LOGICAL NOMBRE-LOGICO

Despliega todos los nombres lógicos en una tabla de nombres lógicos o despliega la equivalencia actual del nombre lógico especificado.

CALIFICADORES IMPORTANTES

/ALL

Indica que todos los nombres lógicos especificados en las tablas de nombres lógicos sean desplegados.

Ejemplos:

$ SHOW LOGICAL SYS$INPUT

Muestra la equivalencia del nombre lógico SYS$INPUT.

$ SHOW LOGICAL /ALL

Muestra todos los nombres lógicos definidos.

.5 COMANDO SHOW MAGTAPE NOMBRE-DE-DISPOSITIVO

Despliega las características y el estado de la unidad de cinta especificada.

Ejemplo:

$SHOW MAGTAPE MTA0:

2.6 COMANDO SHOW MEMORY

Despliega la disponibilidad y utilización de los recursos de memoria.

Ejemplo:

$ SHOW MEMORY

2.7 COMANDO SHOW PRINTER NOMBRE-DE-DISPOSITIVO

Despliega las características definidas para la impresora especificada.

Ejemplo:

$ SHOW PRINTER LPA0:

2.8 COMANDO SHOW PROCESS

Despliega la información referente al proceso en el que se encuentre.

CALIFICADORES IMPORTANES

/ACCOUNTING

Despliega las estadísticas de la sección de terminal.

/ALL

Despliega toda la información disponible, esto es la información de default y la de todos los demás calificadores.

/PRIVILEGES

Despliega los privilegios disponibles por el usuario.

Ejemplo:

$ SHOW PROCESS/ALL

2.9 COMANDO SHOW PROTECTION

Despliega la protección que tendrán todos los archivos que sean creados durante la sesión de terminal ó tarea en BATCH. Esta protección se puede cambiar con el comando SET PROTECTION.

Ejemplo:

SHOW PROTECTION

.10 COMANDO SHOW QUEUE NOMBRE-DE-COLA

Despliega el estado de las tareas metidas en las colas de impresión y BATCH.

CALIFICADORES IMPORTANTES

/ALL

Despliega los nombres de todas las tareas en la cola especificadas.

/BATCH

Despliega los trabajos en la cola de BATCH.

/BRIEF

Despliega la información resumida.

/DEVICE

Despliega el estado de las tareas en todas las colas.

Ejemplo:

$ SHOW QUEUE/DEVICE/ALL

Despliega todas las tareas en todas las colas.

.11 COMANDO SHOW QUOTA

Despliega el espacio en disco autorizado y usado por un usuario.

Ejemplo:

```
$ SHOW QUOTA
```

2.12 COMANDO SHOW STATUS

Despliega el estado del proceso. Este comando no afecta la imagen, se puede continuar su ejecución después de desplegado el estado.

Ejemplo:

```
$ RUN PROG
    .
    .
    .
  ctrl/Y
$ SHOW STATUS
 Despliega el estado de proceso.
$ CONTINUE
 Continua la ejecucion del programa.
```

El comando RUN ejecuta el programa PROG. Mientras el programa está corriendo, con un CTRL/Y es interrumpido y el SHOW STATUS despliega el estado actual. El programa puede continuar con el comando CONTINUE.

2.13 COMANDO SHOW SYMBOL NOMBRE-DE-SIMBOLO

Despliega el valor actual de un símbolo local ó global.

CALIFICADORES IMPORTANTES

/ALL

Despliega todos los valores de los símbolos de la tabla de símbolos que se vayan a indicar. Si no se indica /LOCAL ó /GLOBAL, los símbolos locales serán desplegados.

$ SHOW SYMBOL/GLOBAL/ALL

Despliega todos los símbolos definidos en la tabla de símbolos globales.

2.14 COMANDO SHOW SYSTEM

Despliega una lista de los procesos en el sistema información acerca l estado de cada uno.

Ejemplo:

$ SHOW SYSTEM

2.15 COMANDO SHOW TERMINAL NOMBRE-DE-DISPOSITIVO

Despliega las características de una terminal especificada.

Estas características pueden cambiarse con el comando SET TERMINAL.

Ejemplo:

$ SHOW TERMINAL

Despliega las características de la terminal en la que se encuentre la sesión.

2.16 COMANDO SET DEFAULT NOMBRE-DE-DISPOSITIVO

Cambia el dispositivo y/o directorio de default para el proceso.

El nuevo default es aplicado a todas las subsecuentes especificaciones de archivo que no indiquen dispositivo ó directorio.

Ejemplos:

$ SET DEFAULT DBA2:

Este comando cambia el disco de default a DBA2:; el nombre del directorio de default no cambia.

$ SET DEFAULT [TAREAS]

Este comando cambia el directorio de default a [TAREAS]; el nombre del disco de default no cambia.

$ SET DEFAULT DBA1:[TAREAS.SUB1]

Este comando cambia el disco de default a DBA1: y el directorio de default al subdirectorio [TAREAS.SUB1].

.19 COMANDO SET PROTECTION = CODIGO LISTA-DE-ARCHIVOS

Establece la protección a un archivo ó grupo de archivos.

La protección de un archivo limita el tipo de acceso disponible a otros usuarios del sistema.

Ejemplos:

$ SET PROTECTION=(OWNER,RWE,WORLD:RWE)-

$CASA.LIS, CASA.EXE

Este comando cambia las protecciones de los archivos CASA.LIS y CASA.EXE para el OWNER y el WORLD, las protecciones que tenían para SYSTEM y GROUP no son alteradas.

2.20 COMANDO SET PROTECTION = CODE /DEFAULT

Establece la protección de default de todos los archivos creados, después de dado éste comando y hasta la terminación de la sesión.

Ejemplo:

$ SET PROTECTION = (GROUP:RWED,WOLD:RE) /DEFAULT

2.21 COMANDO SET QUEUE/ENTRY = NUMERO-DE-JOB NOMBRE-DE-COLA

Cambia el estado de un archivo que está en la cola de impresión ó de BATCH pero que todavía no es procesado por el sistema.

LIFICADORES IMPORTANTES

/RELEASE

Libera a un Job del estado Hold en el que había quedado.

Ejemplo:

```
$ PRINT/HOLD  MIARCH.DAT
Job 112 entered on queue  SYS$PRINT
  .
  .
  .
$ SET QUEUE/ENTRY = 112/RELEASE
```

2.22 COMANDO SET TERMINAL

Cambia las características de la terminal.

CALIFICADORES IMPORTANTES

/WIDTH = n

Indica el número de caracteres de cada línea de entrada ó salida.

Ejemplo:

```
$ SET TERMINAL/WIDTH = 132
```

Este comando indica que la longitud de la línea de la terminal es de 132 caracteres.

CURSOS: "INTRODUCCION AL SISTEMA VAX - 11/780" Y
"OPERACION Y ADMINISTRACION DEL SISTEMA VAX /VMS"
DEL 29 DE ABRIL AL 13 DE MAYO.
MEXICO, D.F.

EDITOR EDT MODO LINEA.

PROFESORES:

ING. EDUARDO S. JALLATH CORIA.
ING. ALEJANDRO JIMENEZ GARCIA.
ING. SOCRATES A. MUÑIZ ZAFRA.
ING. HUMBERTO SANCHEZ SANDOVAL.

MAYO DE 1985.

CAPITULO 3

EDITOR EDT MODO LINEA

3.1 INTRODUCCION

Un editor es un programa que nos sirve para generar archivos.

El editor **EDT** puede trabajar de las siguientes formas:

1. Modo linea
2. Modo KEYPAD
3. Modo NOKEYPAD

Para invocar al editor EDT, sólo basta dar un comando de DCL, llamado EDIT. La sintaxis completa del comando EDIT es la siguiente:

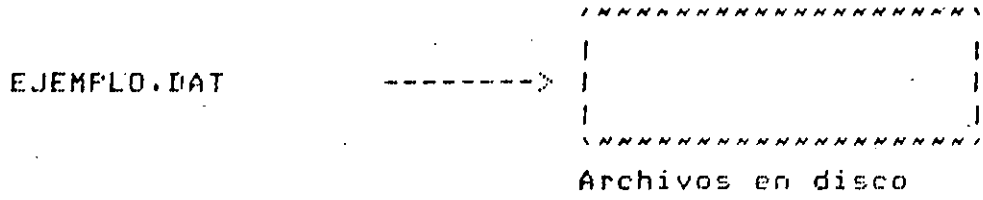
EDIT/EDT <un nombre de archivo>

Al invocar al editor EDT se crea un espacio en memoria, llamado BUFFER, el cual podrá contener información o estar vacío. Esto es: si el archivo que quiero editar ya se encuentra en disco, el buffer va a contener una copia de él; si el archivo no existía en disco, entonces el buffer estará vacío.

Ejemplo de una creación de archivo.

```
$ EDIT EJEMPLO.DAT
Input file does not exist
[EOB]
```

El editor EDT busca en disco el archivo EJEMPLO.DAT.

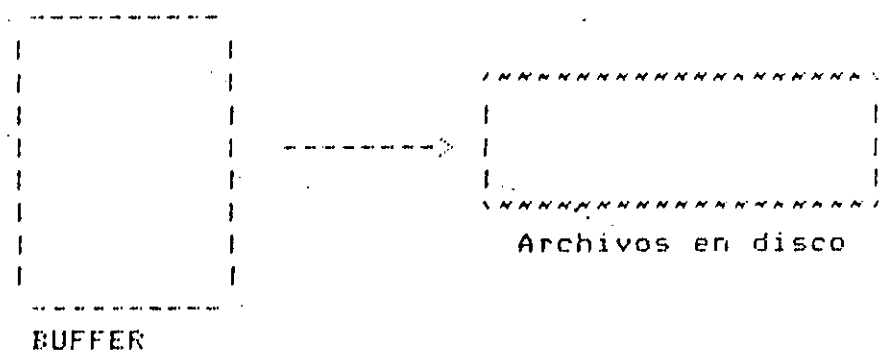


Como no encuentra el archivo en el disco, EDT hace las siguientes acciones:

1. Generá un mensaje

```
Input file does not exist
[EOB]
*
```

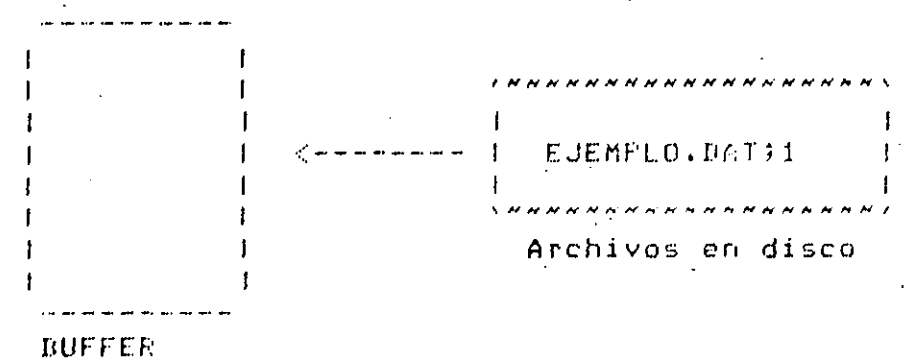
2. Crea un buffer, en el cual se va introducir todo el texto. El momento que nosotros salgamos del editor, se salvará en disco el texto que se introdujo, a menos que destruyamos el buffer.



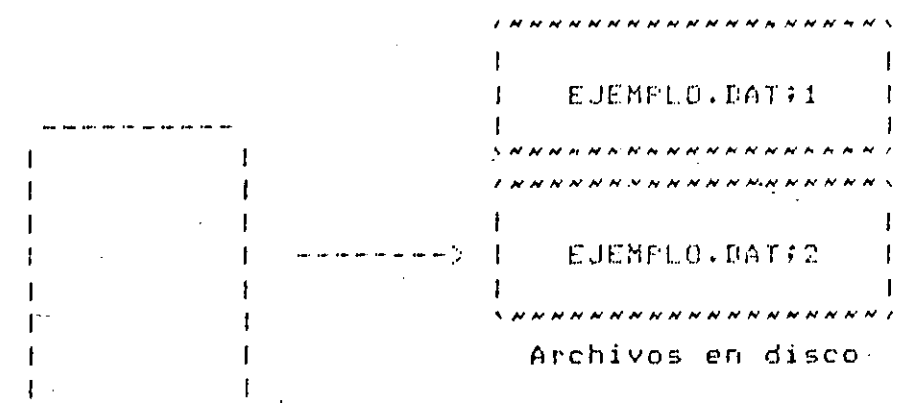
A continuación se da un ejemplo de un archivo que ya existe en disco. En este ejemplo se hará una copia del archivo EJEMPLO.DAT;1 al buffer; En el buffer se harán todas las modificaciones; al finalizar de hacer todas las modificaciones y se proceda a salirse del editor, en ese momento se creará una nueva versión del archivo EJEMPLO.DAT, es decir, se creará el archivo EJEMPLO.DAT;2.

```
$ EDIT EJEMPLO.DAT
      1        Era Damón un filósofo de Siracusa, que fué condenado a  una
*
```

Al invocar a un archivo, el editor buscará primero si existe en disco el archivo que se desea editar, si se encuentra el archivo en disco, como en este caso, se hace una copia al buffer de trabajo, apareciendo en la terminal la primera línea del archivo y en seguida un *.



Al momento que se termina de editar, se hace una copia en disco con una nueva versión, conservándose la versión anterior.



Las características de EDT, son las siguientes:

i. Genera archivos secuenciales.

ii. La secuencia que genera el editor es independiente del archivo.

iii. La secuencia es generada automáticamente.

iv. La secuencia generada tiene incrementos unitarios.

v. La secuencia no es visible al momento de insertar texto.

vi. La secuencia solo es visible cuando se muestra el texto.

vii. Cuando se inserta una línea entre dos ya existentes, la nueva línea adquiere una secuencia decimal.

viii. Al momento de salir del editor, automáticamente, se generá una nueva secuencia unitaria. Es decir, al volver a editar un programa, éste siempre tendrá secuencia unitaria.

ando el control es llevado por el editor, en ese momento deja de aparecer el $ y aparece el *, como símbolo de control del editor (prompt).

Cuando aparece el prompt de EDT (*), en ese momento el editor está listo para recibir y ejecutar cualquier comando de edición.

La sintaxis de la mayoría de los comandos de EDT es la siguiente:

<nombre del comando>/calificador/calificador... [rango]

El nombre de un comando es una palabra reservada, la cual tiene destinada una función específica.

Los calificadores son modificadores del comando.

El rango se refiere al conjunto de líneas, las cuales van hacer afectadas por el comando de edición. Como el rango es una de las partes fundamentales del comando, por esa razón, primero trataremos a todos los rangos que maneja EDT.

El nombre de los comandos y sus calificadores serán explicados ampliamente más adelante.

Para los ejemplo de cada uno de los rangos, así como para los comandos de EDT, se usará el siguiente texto: (El número que aparece en el lado izquierdo es la secuencia que tiene el texto en éste momento.)

```
   1          Era Damón un filósofo de Siracusa, que fué condenado a
   2          muerte por conspirar contra el tirano Dionisio. Este le
   3          concedió permiso para ir a un pueblo cercano a despedirse
   4          de su familia y dejar arreglados sus asuntos, a condición
   5          de que dejase en rehenes un fiador, el cual sufriría la
   6          última pena en lugar de Damón si éste faltaba a su palabra
   7          de acudir a la hora fijada para su ejecución. Su
   8          entrañable amigo, el filósofo Fintias, se ofreció a quedar
   9          como rehén, y como Damón no se presentó a la hora fatal, él
  10          marchó gustoso al patíbulo y desde allí se dirigió al
  11          público diciendo que Damón era inocente de toda culpa; que
  12          tenía la seguridad de que habría sido detenido por alguna
  13          circunstancia, contra su voluntad, y que tal vez en aquel
  14          momento se hallaba en camino para ir á cumplir su palabra;
  15          pero que él, Fintias, no quería que se demorase la
  16          ejecución y moriría con gusto para salvar la vida de su
  17          inocente amigo.
17.1
17.2               Apenas acababa de hablar, oyóse una voz estentórea que
17.3          de lejos gritaba: "¡Deteneos!", y vióse llegar a escape
17.4          tendido un jinete que resultó ser el mismo Damón, el cual,
17.5          subiendo al patíbulo, abrazó a Fintias y le dijo que su
17.6          retraso en llegar era debido a que había reventado su
17.7          caballo y tuvo que pedir otro prestado en el camino para
17.8          poder llegar a tiempo de evitar el sacrificio de su amigo.
17.9
  27               El tirano Dionisio se conmovió tanto al ver el
  28          altruismo de aquellos dos amigos, que no sólo perdonó a
  29          Damón, sino que rogó a ambos que le permitiesen ser
  30          partícipe de su amistad.
[EOB]
```

2 RANGOS QUE MANEJA EDT

El rango es el conjunto de líneas, las cuales van hacer afectadas por el comando. Nosotros definiremos dos tipos de rangos:

1. Los rangos que denotan una sola línea.

2. Los rangos que denotan varias líneas.

Primero definiremos los rangos que denotan una sola línea, ya que los rangos que denotan varias líneas, algunas veces, se definen en función de estos.

3.2.1 Rangos Unitarios

Rangos que denotan una sola línea. En adelante los denotaremos como rangos unitarios

3.2.1.1 Punto

.

Un punto señala la línea donde el cursor se encuentra localizado. Es la línea actual (+).

Ejemplo:

Si se dió el siguiente comando:
COMANDO 12
La línea doce sería afectada por el comando. Ahora la línea doce es la línea actual
. (rango)
12 tenía la seguridad de que habría sido detenido por algunas

3.2.1.2 Nada

Nada señala a la línea actual. Si no ponemos ningún rango, por omisión

Nota: (+) Definiremos a la línea actual, como la última línea mas recientemente usada. En un rango será la primera línea.

omará el ranso '.' (Véase al ranso .).

Ejemplo:

Si se dió el siguiente comando:
COMANDO 9
La línea nueve sería afectada por el comando. Ahora la línea nueve es la línea actual
(ranso)
9 en rehenes, y como Damón no se presentó a la hora fatal, él

Si se usa el comando TYPE, entonces la línea actual será la siguiente línea.

3.2.1.3 Número Entero

<número entero>

Un número entero señala la línea que esté numerada con esa secuencia.

emplo:

5 (ranso)
5 de que dejase en rehenes un fiador, el cual sufriría la

3.2.1.4 Número Decimal

<número decimal>

Un número decimal señala la línea que tiene esa secuencia.

Ejemplo:

17.7 (ranso)
17.7 caballo y tuvo que pedir otro prestado en el camino para

3.2.1.5 'STRING'

'STRING' | "STRING"

Un string encerrado por comas o por comillas sirve para denotar a la próxima línea que contenga al string.

Ejemplos:

(rango)
4 de su familia y dejar arreglados sus asuntos, a condición

Como la línea actual es la cuatro, entonces empezará a buscar a partir de ella, el string amigo.

'amigo' (rango)
8 entrañable amigo, el filósofo Fintias, se ofreció a quedar

Si la línea actual fuese la 17.9. Entonces

. (rango)
17.9
'amigo' (rango)
28 altruismo de aquellos dos amigos, que no sólo le perdonó a

3.2.1.6 -'STRING'

-'STRING' | -"STRING"

Un string precedido por un signo - y encerrado por comas o por comillas, sirve para denotar a la próxima línea anterior que contenga al string.

Ejemplos:
. (rango)
30 participe de su amistad.

Como la línea actual es la treinta, entonce empezará a buscar hacia atrás a partir de ella, al string Damón.

-"Damón" (rango)
29 Damón, sino que rogó a ambos que le permitiesen ser

Si la línea actual fuese la quince. Entonces

. (rango)
15 pero que él, Fintias, no quería que se demorase la
-"Damón" (rango)
11 público diciendo que Damón era inocente de toda culpa; que

Nota: (!) El símbolo | equivale a decir 'ó'

..2.1.7 <Rango Unitario> + <Número Entero>

<rango unitario> + <número entero>

Un rango unitario seguido por el signo + y un número entero, sirve para denotar a la línea que se encuentra n posiciones después del rango unitario. Las n posiciones son definidas por el número entero.

Ejemplo:

15 + 8 (rango)
17.6 retraso en llegar era debido a que había reventado su

3.2.1.8 <Rango Unitario> - <Número Entero>

<rango unitario> - <número entero>

Un rango unitario seguido por el signo - y un número entero, sirve para denotar a la línea que se encuentra n posiciones antes del rango unitario. Las n posiciones son definidas por el número entero.

Ejemplo:

10 - 4 (rango)
6 última pena en lugar de Damón si éste faltaba a su palabra

3.2.1.9 BEGIN

BEGIN

La palabra BEGIN sirve para denotar la primera línea del buffer.

Ejemplo:

BEGIN (rango)
1 Era Damón un filósofo de Siracusa, que fué condenado a una

3.2.1.10 END

END

La palabra END sirve para denotar a la última línea del buffer (+). Esta siempre estará vacía.

Ejemplo:

```
END (ranso)
[EOB]
```

3.2.1.11 LAST

LAST

La palabra LAST sirve para denotar la última línea usada en un buffer, esto es, denota a la que fué la línea actual en el buffer pasado. (Se utiliza en manejo de buffers)

3.2.1.12 ORIGINAL

ORIGINAL (<número entero> | < número decimal>)

La palabra ORIGINAL seguida de un número entero sirve para denotar a la línea que tenía originalmente la secuencia dada por el número, antes de haber sido resecuenciado el buffer. En otras palabras, hace referencia a la línea que tenía antes esa secuencia en el buffer, aunque ahora tenga otra secuencia.

Nota: (+) La última línea de un buffer, siempre es una línea en blanco. La justificación de que sea una línea en blanco, será dada cuando se vea el comando INSERT.

Ejemplo:

Si suponemos que originalmente el archivo tenía una secuencia unitaria.

```
ORIGINAL 19 (rango)
    17.2                Apenas acababa de hablar, oúóse una voz estentórea que
```

3.2.2 Rangos Múltiples

Rangos que denotan a un conjunto de líneas. En adelante los denotaremos simplemente rangos.

3.2.2.1 <rango Unitario Inferior> : <rango Unitario Superior>

<rango unitario inferior> : <rango unitario superior>

Un rango unitario inferior seguido de : y del rango unitario superior, sirve para denotar el conjunto de líneas, limitadas por los rangos unitarios, incluyéndolos. El rango unitario inferior deberá ser menor que rango unitario superior.

Ejemplo:

```
17.9:END (rango)
    17.9
    27                  El tirano Dionisio se conmovió tanto al ver el
    28              altruísmo de aquellos dos amigos, que no sólo le perdonó a
    29              Damón, sino que rogó a ambos que le permitiesen ser
    30              partícipe de su amistad.
[EOB]
```

3.2.2.2 <rango Unitario Inferior> THRU <rango Unitario Superior>

<rango unitario inferior> THRU <rango unitario superior>

El rango unitario inferior seguido de la palabra THRU y del rango unitario superior, sirve para denotar el conjunto de líneas, limitadas por los rangos unitarios, incluyéndolos.El rango unitario inferior deberá ser menor que el rango unitario superior.

Ejemplo:

```
_GIN THRU 5 (rango)
     1              Era Damón un filósofo de Siracusa, que fué condenado a una
     2              muerte por conspirar contra el tirano Dionisio. Este le
     3              concedió permiso para ir a un pueblo cercano a despedirse
```

```
    4           de  su  familia y dejar arreglados sus asuntos, a condición
    5           de que dejase en rehenes un fiador,  el  cual  sufriría  la
```

3.2.2.3 <rango Unitario # <número Entero>

<rango unitario> # <número entero>

El rango unitario seguido de # y de un número entero, sirve para denotar n líneas apartir del rango unitario. Las n líneas son dadas por el número entero.

Ejemplo:

```
16#6 (rango)
   16           ejecución  y  moriría  con  gusto para salvar la vida de s
   17           inocente amigo.
   17.1
   17.2                Apenas acababa de hablar, oúose una voz estentórea que
   17.3           de  lejos  gritaba:  '¡Deteneos!', y viósc llegar a galop
   17.4           tendido un jinete que resultó ser el mismo Damón, el  cual
```

3.2.2.4 BEFORE

BEFORE

La palabra BEFORE sirve para denotar a todas las líneas que se encuentran antes de la línea actual.

Ejemplos:

```
. (rango)
    5           de que dejase en rehenes un fiador,  el  cual  sufriría  la
```

Como la línea actual es la cinco. Entonces

```
BEFORE (rango)
    1           Era Damón un filósofo de Siracusa, que fué condenado a
    2           muerte  por  conspirar  contra el tirano Dionisio. Este le
    3           concedió permiso para ir a un pueblo cercano  a  despedirse
    4           de  su  familia y dejar arreglados sus asuntos, a condición
```

2.2.5 REST

REST

La palabra REST sirve para denotar a todas las líneas que se encuentran desde la línea actual hasta el fin del buffer.

Ejemplos:

```
* (rango)
   27                El tirano Dionisio se conmovió tanto al ver el
```

Como la línea actual es la veintisiete. Entonce

```
REST (rango)
   27                El tirano Dionisio se conmovió tanto al ver el
   28           altruismo de aquellos dos amigos, que no sólo le perdonó a
   29           Damón, sino que rogó a ambos que le permitiesen ser
   30           partícipe de su amistad.
[EOB]
```

3.2.2.6 WHOLE

WHOLE

La palabra WHOLE sirve para denotar a todas las líneas del buffer.

Ejemplo:

```
WHOLE (rango)
    1           Era Damón un filósofo de Siracusa, que fué condenado a
    2           muerte por conspirar contra el tirano Dionisio. Este le
    3           concedió permiso para ir a un pueblo cercano a despedirse
    4           de su familia y dejar arreglados sus asuntos, a condición
    5           de que dejase como rehén un fiador, el cual sufriría la
    6           última pena en lugar de Damón si éste faltaba a su palabra
    7           de acudir a la hora fijada para su ejecución. Su
    8           entrañable amigo, el filósofo Fintias, se ofreció a quedar
    9           en rehén, y como Damón no se presentó a la hora fatal, él
   10           marchó gustoso al patíbulo y desde allí se dirigió al
   11           público diciendo que Damón era inocente de toda culpa; que
   12           tenía la seguridad de que habría sido detenido por alguna
   13           circunstancia, contra su voluntad, y que tal vez en aquel
   14           momento se hallaba en camino para ir a cumplir su palabra;
   15           pero que él, Fintias, no quería que se demorase la
   16           ejecución y moriría con gusto para salvar la vida de su
   17           inocente amigo.
   17.1
   17.2                Apenas acababa de hablar, oyóse una voz estentórea que
   17.3           de lejos gritaba: "¡Deteneos!", y vióse llegar a galope
   17.4           tendido un jinete que resultó ser el mismo Damón, el cual,
   17.5           subiendo al patíbulo, abrazó a Fintias y le dijo que su
   17.6           retraso en llegar era debido a que había reventado su
   17.7           caballo y tuvo que pedir otro prestado en el camino para
   17.8           poder llegar a tiempo de evitar el sacrificio de su amigo.
   17.9
   27                El tirano Dionisio se conmovió tanto al ver el
```

```
   28          altruismo  de aquellos dos amigos, que no sólo perdonó a
   29          Damón, sino que rogó a ambos  que  le  permitiesen  ser
   30          partícipe de su amistad.
[EOB]
```

3.2.2.7 <rango Unitario>,<rango Unitario>,<rango Unitario>,,,

<rango unitario>,<rango unitario>,<rango unitario>...

La lista de rangos unitarios sirve para denotar al conjunto de líneas especificadas por los rangos unitarios. Los rangos unitarios pueden ir en cualquier orden, esto es, pueden ser de mayor a menor, salteados, de menor a mayor o la combinación de los anteriores.

Ejemplo:

```
13,14,3,17.6 (rango)
   13          circunstancia, contra su voluntad, y que tal vez en aquel
   14          momento  se hallaba en camino para ir a cumplir su palabra
    3          concedió permiso para ir a un pueblo cercano a despedirse
   17.6        retraso en llegar era  debido  a  que  había  reventado  su
```

3.2.2.8 ALL 'STRING'

[<rango>] ALL 'STRING'

El rango seguido de la palabra ALL y un string encerrado entre comillas o apóstrofes, sirve para denotar el conjunto de líneas que contienen el string, dentro del rango especificado. Si se omite el rango se asumirá todo el buffer.

Ejemplo:

```
6:17 ALL "Fintias" (rango)
    8          entrañable amigo, el filósofo Fintias, se ofreció a quedar
   15          pero que él, Fintias, no quería que se demorase la
```

A continuación, describiremos los comandos más usados. Todos los comandos tienen más o menos la misma importancia, por esa razón los daremos en orden alfabético. La parte más obscura es la mínima abreviación, para que el editor interprete de que tipo de comando se trata.

Los comandos de EDT son palabras en inglés, cada palabra significa lo que el comando hace, por tal motivo muchas veces será redundante la explicación. Un ejemplo de lo anterior sería el comando COPY, cuya traducción es copia, que es la función del comando.

El texto quedará como estaba originalmente después de desmostrar cada comando. Esto es, no importa el orden en que se vean los comandos de EDT, ya que el texto, después de ilustrar al comando, quedará igual.

3.3 COMANDOS Y CALIFICADORES

3.3.1 COPY

COPY [<rango origen>] TO [<rango unitario destino] [/QUERY] [/DUPLICATE:n]

El comando **COPY** sirve para hacer una copia de todo el rango origen, (o rango unitario origen) al rango unitario destino. El texto quedará tanto en el rando origen, como inmediatamente antes del rando destino. Si se omite algún rango, se asumirá a la línea actual como rango.

El rango destino puede estar incluido en el rango origen.

Después de efectuar el comando, el rango unitario destino será la línea actual.

Los calificadores del comando copy causarán los siguientes efectos:

1. El calificador QUERY sirve para verificar cada línea que se desee copiar. Para hacerlo, preguntará línea por línea, a lo cual podremos responder, inmediatamente después que aparezca el prompt del query (?), lo siguiente:

   Y (yes), si se desea que se copie esa línea.

   N (no), si no se deseo que se copie esa línea.

   A (all), si se desea que se copien todas las líneas siguientes. Deja de preguntar para las demás líneas. Equivale a responder S, siempre que aparece el ?.

   Q (quit), Deja de copiar todas las demás líneas. Equivale a responder N, siempre que aparece el ?.

2. El calificador DUPLICATE:n sirve para que se copie, tantas veces como indica la n, el rango origen. Esto es, equivale a hacer n copys iguales.

   Si se omite :n, no se copia el rango origen. Esto es, equivale a no haber hecho el copy.

   Si se omite el calificador DUPLICATE, o se pone DUPLICATE:1, se efectua la misma acción.

Ejemplos:

```
*COPY 1:3 TO 10
2 lines copied
*
```

```
    1           Era Damón un filósofo de Siracusa, que fué condenado a
    2           muerte por conspirar contra el tirano Dionisio. Este le
    3           concedió permiso para ir a un pueblo cercano a despedirse
    4           de su familia y dejar arreglados sus asuntos, a condición
    5           de que dejase como rehén un fiador, el cual sufriría la
    6           última pena en lugar de Damón si éste faltaba a su palabra
    7           de acudir a la hora fijada para su ejecución. Su
    8           entrañable amigo, el filósofo Fintias, se ofreció a quedar
    9           como rehén, y como Damón no se presentó a la hora fatal, él
    9.1         Era Damón un filósofo de Siracusa, que fué condenado a
    9.2         muerte por conspirar contra el tirano Dionisio. Este le
    9.3         concedió permiso para ir a un pueblo cercano a despedirse
   10           marcho gustoso al patíbulo y desde allí se dirigió al
   11           público diciendo que Damón era inocente de todo cuipa: que
   12           tenía la seguridad de que habría sido detenido por alguna
   13           circunstancia, contra su voluntad, y que tal vez en aquel
    .
    .
    .
   29           Damón, sino que rogó a ambos que le permitiesen ser
   30           partícipe de su amistad.
[EOB]
```

```
*COPY 17.1#4 TO 17.3
4 lines copied
*
```

```
    .
    .
    .
   17           inocente amigo.
   17.1
   17.2               Apenas acababa de hablar, oyóse una voz estentórea que
   17.21
   17.22              Apenas acababa de hablar, oyóse una voz estentórea que
   17.23       de lejos gritaba: "¡Deteneos!", y vióse llegar a galope
   17.24       tendido un jinete que resultó ser el mismo Damón, el cual
   17.3        de lejos gritaba: "¡Deteneos!", y vióse llegar a galope
   17.4        tendido un jinete que resultó ser el mismo Damón, el cual
   17.5        subiendo al patíbulo, abrazó a Fintias y le dijo que su
   17.6        retraso en llegar era debido a que había reventado su
   17.7        caballo y tuvo que pedir otro prestado en el camino para
   17.8        poder llegar a tiempo de evitar el sacrificio de su amigo.
   17.9
    .
    .
```

```
   29          Damón, sino que rogó a ambos que la permitiesen ser
   30          partícipe de su amistad.
[EOB]
```

```
*COPY ALL "Damón" TO BEGIN/QUERY

    1           Era Damón un filósofo de Siracusa, que fué condenado a
?n
    6           última pena en lugar de Damón si éste faltaba a su p . . a
?w
    9           como rehén, y como Damón no se presentó a la hora fatal, él
?a
1 line copied
*

    0.1         última pena en lugar de Damón si éste faltaba a su palabra
    1           Era Damón un filósofo de Siracusa, que fué condenado a una
    2           muerte por conspirar contra el tirano Dionisio. Este le
    3           concedió permiso para ir a un pueblo cercano a despedirse
    .                 .
    .
    .
   29           Damón, sino que rogó a ambos que le permitiesen ser
   30           partícipe de su amistad.
[EOB]

COPY S,10 TO END/DUPLICATE:3
  lines copied 3 times

    .
    .
    .
   29           Damón, sino que rogó a ambos que le permitiesen ser
   30           partícipe de su amistad.
   31           entrañable amigo, el filósofo Fintias, se ofreció a quedar
   32           marchó gustoso al patíbulo y desde allí se dirigió al
   33           entrañable amigo, el filósofo Fintias, se ofreció a quedar
   34           marchó gustoso al patíbulo y desde allí se dirigió al
   35           entrañable amigo, el filósofo Fintias, se ofreció a quedar
   36           marchó gustoso al patíbulo y desde allí se dirigió al
[EOB]
```

3.3.2 DELETE

DELETE [<rango>] [/QUERY]

El comando **DELETE** sirve para borrar todas las líneas especificadas en el rango. Si se omite el rango, se asumirá la línea actual como rango. Después de borrar el conjunto de líneas, hará lo siguiente:

aparecerá un mensaje, n line deleted, en donde n, es un número entero, que indica el total de líneas borradas; posteriormente aparecerá la línea siguiente de la última línea borrada la cual será la línea actual.

El calificador QUERY sirve para verificar cada línea que se desee borrar para hacerlo preguntará línea por línea, a lo cual nosotros le podremos responder, inmediatamente después que aparezca el prompt del query (?), lo siguiente:

Y (yes), si se desea que se borre esa línea.

N (no), si no se desea que se borre esa línea.

A (all), si se desea que se borren todas las líneas siguientes. Deja de preguntar para las demás líneas. Equivale a responder S, siempre que aparece el ?.

Q (quit), Deja de borrar todas las demás líneas. Equivale a responder N, siempre que aparece el ?.

Ejemplos:

```
. (rango)
   10           marchó sustoso al patíbulo y desde allí se dirisió al

*DELETE -"Fintias"
1 line deleted
    9           como rehén, y como Damón no se presentó a la hora fatal, él
*
    .
    .
    .
    6           última pena en lugar de Damón si éste faltaba a su palabra
    7           de acudir a la hora fijada para su ejecución. Su
    9           como rehén y como Damón no se presentó a la hora fatal, él
   10           marchó sustoso al patíbulo y desde allí se dirisió al
   11           público diciendo que Damón era inocente de toda culpa; que
    .
    .
    .
   29           Damón, sino que rogó a ambos que le permitiesen ser
   30           partícipe de su amistad.
  [OEF]

*DELETE 3,2,9
3 lines deleted
   10           marchó sustoso al patíbulo y desde allí se dirisió al
```

```
    1           Era Damón un filósofo de Siracusa, que fué condenado a
    4           de su familia y dejar arreglados sus asuntos, a condición
    5           de que dejase en rehenes un fiador, el cual sufriría la
    6           última pena en lugar de Damón si éste faltaba a su palabra
    7           de acudir a la hora fijada para su ejecución.  Su
    8           entrañable amigo, el filósofo Fintias, se ofrecio a quedar
   10           marchó gustoso al patíbulo y desde allí se dirigió al
   11           público diciendo que Damón era inocente de toda culpa; que
                .
                .
                .
   29           Damón, sino que rogó a ambos que le permitiesen ser
   30           partícipe de su amistad.
[EOB]

*DELETE 17.1 THRU 17.9/QUERY

   17.1
?y
   17.2             Apenas acababa de hablar, oyóse una voz estentórea que
?n
   17.3          de lejos gritaba:  "¡Deteneos!", y viése llegar a escape

 9 lines deleted
   27             El tirano Dionisio se conmovió tanto al ver el
*
                .
                .
                .
   15           pero que él, Fintias, no quería que se demorase la
   16           ejecución y moriría con gusto para salvar la vida de su
   17           inocente amigo.
   17.2             Apenas acababa de hablar, oyóse una voz estentórea que
   27               El tirano Dionisio se conmovió tanto al ver el
   28           altruísmo de aquellos dos amigos, que no sólo le perdonó a
   29           Damón, sino que rogó a ambos que le permitiesen ser
   30           partícipe de su amistad.
[EOB]
```

3.3.3 EXIT

EXIT [<nombre de archivo>]

El comando **EXIT** sirve para finalizar una sesión de edición. Al terminar la sesión, se copia el buffer de trabajo, con todas sus modificaciones, al disco. Si no se pone un nombre de archivo después del comando, se salvará con el mismo nombre que se le dió al momento de darle EDIT; Si se le da un nombre, entoces se salvará en disco con ese nombre.

N (no), si no se desea que se mueva esa línea.

A (all), si se desea que se muevan todas las líeas siguientes. Deja de preguntar para los demás líneas. Equivale a responder Y, siempre que aparece el ?.

Q (quit), Deja de mover todas las demás líneas. Equivale a responder N, siempre que aparece el ?.

Ejemplos:

```
*MOVE 2:5 TO 30
4 lines moved
*
    1           Era Damón un filósofo de Siracusa, que fué condenado a
    6           última pena en lugar de Damón si éste faltaba a su palabra
    7           de acudir a la hora fijada para su ejecución.  Su
    .
    .
    .
   29           Damón, sino que rogó a ambos que le permitiesen ser
   29.1         muerte por conspirar contra el tirano Dionisio. Este le
   29.2         concedió permiso para ir a un pueblo cercano a despedirse
   29.3         de su familia y dejar arreglados sus asuntos, a condición
   29.4         de que dejase en rehenes un fiador, el cual sufriría la
   30           partícipe de su amistad.
[EOB]
```

```
cecafi

*MOVE 10#5 TO BEGIN/QUERY

   10          marchó gustoso al patíbulo y desde allí se dirisió al
?n
   11          público diciendo que Damón era inocente de toda culpa; que
?n
   12          tenía la seguridad de que habría sido detenido por alguna
?a
4 line moved
*
    0.1        público diciendo que Damón era inocente de toda culpa; que
    0.2        tenía la seguridad de que habría sido detenido por alguna
    0.3        circunstancia, contra su voluntad, y que tal vez en aquel
    0.4        momento se hallaba en camino para ir a cumplir su palabra;
    1          Era Damón un filósofo de Siracusa, que fué condenado a
    2          muerte por conspirar contra el tirano Dionisio. Este le
    3          concedió permiso para ir a un pueblo cercano a despedirse
    4          de su familia y dejar arreglados sus asuntos, a condición
    5          de que dejase como rehé un fiador, el cual sufriría la
    6          última pena en lugar de Damón si éste faltaba a su palabra
    7          de acudir a la hora fijada para su ejecución. Su
    8          entrañable amigo, el filósofo Fintias, se ofreció a quedar
    9          como rehé, y como Damón no se presentó a la hora fatal, él
   10          marchó gustoso al patíbulo y desde allí se dirisió al
   15          pero que él, Fintias, no quería que se demorase la
   16          ejecución y moriría con gusto para salvar la vida de su
   17          inocente amigo.
   17.1
   17.2              Apenas acababa de hablar, oyóse una voz estentórea que
    .
    .
    .
   29          Damón, sino que rogó a ambos que le permitiesen ser
   30          partícipe de su amistad.
[EOB]
```

3.3.9 PRINT

PRINT <nombre de archivo> [rango]

El comando PRINT sirve para escribir texto del buffer de trabajo a un archivo en disco. El texto quedará delimitado por el rango.

Las características que tendrá el archivo en disco son las siguientes:

1. Tendrá el nombre que se le dió en el comando print.

2. El archivo contendrá las secuencias, que tenía el buffer al momento de hacer el PRINT.

Si se omite el rango, se escribirá todo el buffer de trabajo.

El comando **PRINT** puede ser muy útil cuando escribimos un programa en BASIC por ejemplo, y se nos olvide ponerle secuencia. Con el comando print podemos solucionar ese problema ya que podemos resecuenciar las líneas del buffer y mandar escribir el buffer como un archivo con la secuencia del editor.

Ejemplo:

Si tecleamos el siguiente programa en BASIC:

```
1           REM Este programa no tiene secuencia, por lo que
2           REM al compilar marcará error.
3           PRINT "Pasaba una abejita..."
            PRINT "SHUTHHH..."
            END
^       `---------- programa ----------'
secuencia
del editor
```

Este programa no puede compilar, para corregir éste error se usa el comando PRINT

```
*PRINT ABEJITA.BAS
*
```

El archivo quedará de la siguiente forma:

```
1           1       REM Este programa no tiene secuencia, por lo que
2           2       REM al compilar marcará error.
3           3       PRINT "Pasaba una abejita..."
4           4       PRINT "SHUTHHH..."
5           5       END
^       `---------- programa ----------'
secuencia
del editor
```

Ahora ya se puede compilar.

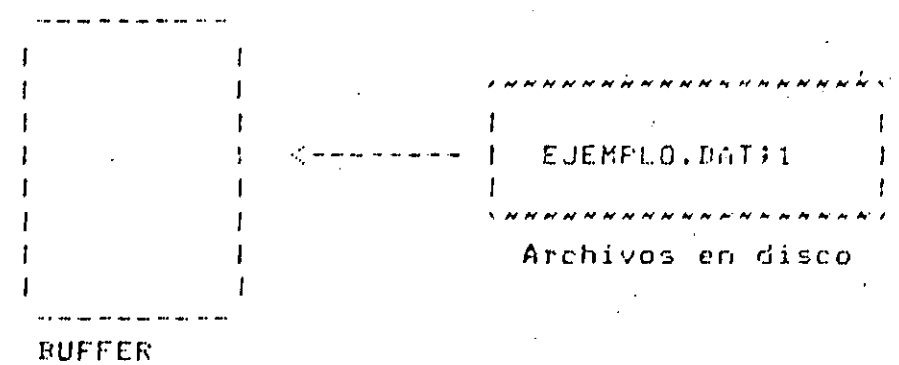
3.3.10 QUIT

QUIT

El comando QUIT sirve para finalizar una sesión de edición, sin salvar el buffer de trabajo en disco. El comando QUIT equivale a no haber entrado a la sesión de edición.

Ejemplo:

```
$ EDIT EJEMPLO.DAT
      1       Era Damón un filósofo de Siracusa, que fué condenado a  una
*
```



Si se dieron los siguientes comandos:

COMANDOa 13

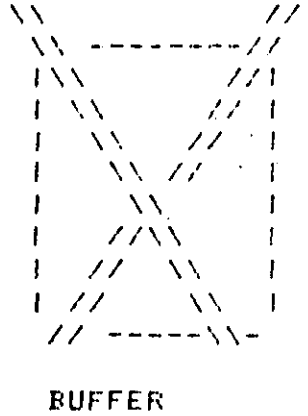
La línea trece sería afectada por el comando.

Ejemplo:

COMANDOb ransox

La línea x sería afectada por el comando.

```
*QUIT
$
```



El buffer se destruye, el archivo EJEMPLO.DAT queda como estaba antes de entrar al editor. El dar QUIT equivale a no entrar a edición.

3.3.11 REPLACE

REPLACE [<rango>] [<;línea a ser insertada>]

El comando REPLACE\ sirve para borrar las líneas especificadas en el rango, e inserta líneas antes de la primera línea del rango que borro. El comando REPLACE es equivalente a un DELETE [<rango>] seguido de un INSERT [;Línea a ser insertada].

Después de borrar el rango de líneas, hará lo siguiente: aparecerá un mensaje, n line deleted, en donde n, es un número entero, que indica el total de líneas borradas; se inserta el texto; posteriormente aparecerá la línea siguiente de la última línea borrada.

Cuando sólo se desee remplazar una línea, basta dar el comando seguido del rango e inmediatamente después un ;, el cual indicara que todo lo que siga hasta el RETURN, será sustituido por lo del rango. La línea se va a insertar antes del rango.

Si omitimos el rango, se asumirá la línea actual como rango.

Ejemplos:

```
*REPLACE 17.2;17.9;"Antes de confiar en un amigo, ponlo a prueba."
8 lines deleted
   27                El  tirano  Dionisio  se  conmovió  tanto  al  ver  el
*
    .
    .
    .
   15           pero que él, Fintias, no quería que se demorase la
   16           ejecución  y  moriría  con  gusto para salvar la vida de su
   17           inocente amigo.
   17.1
   17.11       "Antes de confiar en un amigo, ponlo a prueba."
   27                El  tirano  Dionisio  se  conmovió  tanto  al  ver  el
   28           altruismo  de  aquellos dos amigos, que no sólo le perdonó a
   29           Damón, sino que rogó a ambos que le permitiesen ser
   30           partícipe de su amistad.
[EOB]

*REPLACE 27;4
4 lines deleted
*
            Como el oro se prueba con el fuego, así la felicidad de un
 migo
         se conoce en la adversidad. -Ovidio^Z
[EOB]
*
```

```
      .
      .
      .
   17.992                si al morir, alguien me llore,
   17.993                ¡yo no habré vivido en vano!
   17.994
   18              Como el oro se prueba con el fuego, así la felicidad de
un amigo
   19              se conoce en la adversidad. -Ovidio^Z
[EOB]
```

3.3.12 RESEQUENCE

RESEQUENCE [<rango>] [/SEQUENCE [:inicio [:incremento]]]

El comando RESEQUENCE sirve para cambiar al buffer la secuencia. Si se usan los calificadores, la secuencia empezará en uno y se incrementa de uno en uno.

emplos:

```
*RESEQUENCE 17.1:17.9
Range specified by /SEQUENCE would cause duplicate or non-sequential
numbers
*
```

El archivo no se pudo resecuenciar.

Cuando no se pone el calificador /SEQUENCE los incrementos serán unitarios. Si tenemos la línea 17.1 la resecuenciará como la línea 18, si tenemos la línea 17.2 y la resecuenciamos, le asignará el número 18. Por la

> El calificador SEQUENCE sirve para asignar la secuencia de inicio, así como el incremento de las secuencias.

Si omitimos el rango, se resecuenciará a todo el buffer. razón anterior es que no se pudo resecuenciar

```
:ESEQUENCE
30 lines resequenced
*
```

1 Era Damón un filósofo de Siracusa, que fué condenado a uno

```
    2           muerte por conspirar contra el tirano Dionisio. Este le
                .
                .
                .
   16           ejecución y moriría con gusto para salvar la vida de su
   17           inocente amigo.
   18
   19                Apenas acababa de hablar, oyóse una voz estentórea que
   20           de lejos gritaba: "¡Deteneos!", y vióse llegar a galope
   21           tendido un jinete que resultó ser el mismo Damón, el cual,
   22           subiendo al patíbulo, abrazó a Fintias y le dijo que su
   23           retraso en llegar era debido a que había reventado su
   24           caballo y tuvo que pedir otro prestado en el camino para
   25           poder llegar a tiempo de evitar el sacrificio de su amigo.
   26
   27                El tirano Dionisio se conmovió tanto al ver el
   28           altruismo de aquellos dos amigos, que no sólo le perdonó a
   29           Damón, sino que rogó a ambos que le permitiesen ser
   30           partícipe de su amistad.
[EOB]

*RESEQUENCE/SEQ:100:10
30 lines resequenced
*
  100           Era Damón un filósofo de Siracusa, que fué condenado a una
  110           muerte por conspirar contra el tirano Dionisio. Este le
                .
                .
                .
  250           ejecución y moriría con gusto para salvar la vida de su
  260           inocente amigo.
  270
  280                Apenas acababa de hablar, oyóse una voz estentórea que
  290           de lejos gritaba: "¡Deteneos!", y vióse llegar a galope
  300           tendido un jinete que resultó ser el mismo Damón, el cual,
  310           subiendo al patíbulo, abrazó a Fintias y le dijo que su
  320           retraso en llegar era debido a que había reventado su
  330           caballo y tuvo que pedir otro prestado en el camino para
  340           poder llegar a tiempo de evitar el sacrificio de su amigo.
  350
  360                El tirano Dionisio se conmovió tanto al ver el
  370           altruismo de aquellos dos amigos, que no sólo le perdonó a
  380           Damón, sino que rogó a ambos que le permitiesen ser
  390           partícipe de su amistad.
[EOB]
```

3.13 SUBSTITUTE

SUBSTITUTE /<string original>/<nuevo string>/ [<rango>] [/BRIEF[:n]] [/QUERY] [/NOTYPE]

El comando SUBSTITUTE sirve para cambiar todos los string original que encuentre en el rango especificado, por el nuevo string. El tamaño de los strings puede ser de 0 a 64 caracteres. Después de hacer los cambios en el rango, hará lo siguiente:

1. Si el string original se encontraba en el rango especificado.

   Aparecerán las líneas en donde se hicieron los cambios. Las líneas se, mostraran con los cambios ya hechos; posteriormente aparecerá un mensaje, n substitution made, en donde n, es un número entero, que indica el total de modificaciones hechas.

2. Si el string original no se encuentra en el rango especificado. Aparecerá un mensaje, No substitutions, el cual indicará que no se hizo ninguna modificación.

   Cuando se omite el rango, se cambiará solo la primera ocurrencia del string original, que se encuentre en la línea actual.

   El delimitador / puede ser cualquier caracter. Siempre que no se utilice un caracter alfanumérico y se utilice el mismo delimitador a lo largo de todo el comando. (Los delimitadores pueden ser * & ^ # etcetera.)

El calificador BRIEF sirve para que no se desplieguen todos los caracteres de la línea en donde se hicieron los cambios. Si nosotros omitimos los :n, se desplegarán los primeros 10 caracteres de la línea donde se hicieron los cambios, si deseamos que solo aparezcan n caracteres, agregamos :n donde n, será un número entero que indicará los caracteres que se mostrarán.

De ninguna manera el BRIEF delimita la zona donde se van hacer las sustituciones, solo sirve para que no se liste toda la línea.

El calificador QUERY sirve para verificar cada ocurrencia del string original que se desee modificar. Para hacerlo, preguntará ocurrencia por ocurrencia del string original, a lo cual le podremos responder, inmediatamente después que aparezca el promt

del query (?), lo siguiente:

Y (yes), si se desea que se modifique esa ocurrencia.

N (no), si no se desea que se modifique esa ocurrencia.

A (all), si se desea que se modifiquen todas las ocurrencias siguientes. Deja de preguntar para las demás ocurrencias. Equivale a responder S, siempre que aparece el ?.

Q (quit), Deja de modificar a todas las demás ocurrencias. Equivale a responder N, siempre que aparece el ?.

El calificador NOTYPE sirve para que no se muestren las líneas donde se hicieron las modificaciones.

Ejemplos:

```
  UBSTITUTE/A/E/WHOLE
    1             ErE DEmón un filósofo de SirEcusE, que fué condenEdo E  un
7 substtitutions
*

Si la línea actual es la siguiente:
.
    1             Era Damón un filósofo de Siracusa, que fué condenado a  una

*SUBSTITUTE/A/E/
    1             ErE Damón un filósofo de Siracusa, que fué condenado a  una
1 substtitutions
*

*SUBSTITUTE/entreambos/los dos/whole/BRIEF:12
   29             Damón, sino
*

   29             Damón, sino que rosó a los dos  que  le  permitiesen  ser
*SUBSTITUTE/que/el cual/ ALL ',"/QUERY
    5                1           Era Damón un filósofo de Siracusa, que fué
 condenado a  una
?y
    5                1           Era Damón un filósofo de Siracusa, el cual
f
  condenado a  una
    9                5           de que dejase como rehén un fiador,  el
cual  s
ufriría  la
?n
```

```
   12                8          entrañable amigo, el filósofo Fintias, se
ofr
ecio a quedar
?a
1 substitution
*

*SUBSTITUTE/tirano/malvado/2/NOTYPE
1 substitution
*

    2           muerte  por  conspirar  contra el malvado Dionisio.  Este
le
```

3.3.14 SUBSTITUTE NEXT

SUBSTITUTE NEXT [/<string original>/<nuevo string>/]

El comando SUSTITUTE NEXT sirve para sustituir a la siguiente ocurrencia l string original, por el nuevo string.

Si se omiten los string, se asumirán los strings que hayan tenido, en el último comando SUBSTITUTE. Si no se había usado el SUBSTITUTE, no realizará ninguna función el comando.

Ejemplos:

```
*SUBSTITUTE/era/fué/1
    1           fué Damón un filósofo de Siracusa, que fué condenado a  una
1 substitution
*
*NEXT
   11           público diciendo que Damón fué inocente de toda culpa;  que
*
```

3.3.15 TYPE

TYPE [<rango>] [/BRIEF [: n]] [/STAY]

El comando TYPE sirve para mostrarnos líneas de un texto, las líneas a ostrar estarán delimitadas por el rango. Si no ponemos el comando TYPE, se asumirá éste, esto quiere decir que lo mínimo que podemos escribir del comando es nada, cuando no se ponga nada del comando, entonces se tendrá que usar el %, para cuando se use un rango que empiece con letra, (pasado

el rango).

Si omitimos el rango, se asumirá a la línea actual como rango.

El calificador BRIEF sirve para que no se despliegen todos los caracteres del rango seleccionado. Si nosotros omitimos los :n, se desplegarán los primeros 10 caracteres, si deseamos que sólo aparezcan n caracteres, agregamos :n donde n, será un número entero que indicará los caracteres que se mostrarán.

El calificador STAY sirve para que no cambie de lugar la línea actual, después de ejecutar el comando. La línea actual será la misma que tenía antes de hacer el comando.

Ejemplos:

```
*TYPE 10:13
   10           marchó gustoso al patíbulo y desde allí se dirigió al
   11           público diciendo que Damón fué inocente de toda culpa; que
   12           tenía la seguridad de que habría sido detenido por alguna

*3
    3           concedió permiso para ir a un pueblo cercano a despedirse
*
*BEGIN
    1           Era Damón un filósofo de Siracusa, que fué condenado a
*

*TYPE 17.2:17.5/BRIEF:5
   17.2
   17.3         de  l
   17.4         tendi
   17.5         subie
*

Si, la línea actual es en éste momento la cinco
*.
    5           de que dejase como rehén un fiador, el cual sufriría la
*
*TYPE 3
    3           concedió permiso para ir a un pueblo cercano a despedirse
*

La línea actual será la tres
*TYPE 15/STAY

   15           pero que él, Fintias, no quería que se demorase la
*

La línea actual seguirá siendo la tres.
```

3.3.16 WRITE

WRITE [<nombre de archivo>] [<rango>]

El comando WRITE sirve para escribir en disco una copia del rango especificado, con el nombre que se le de a continuación del comando.

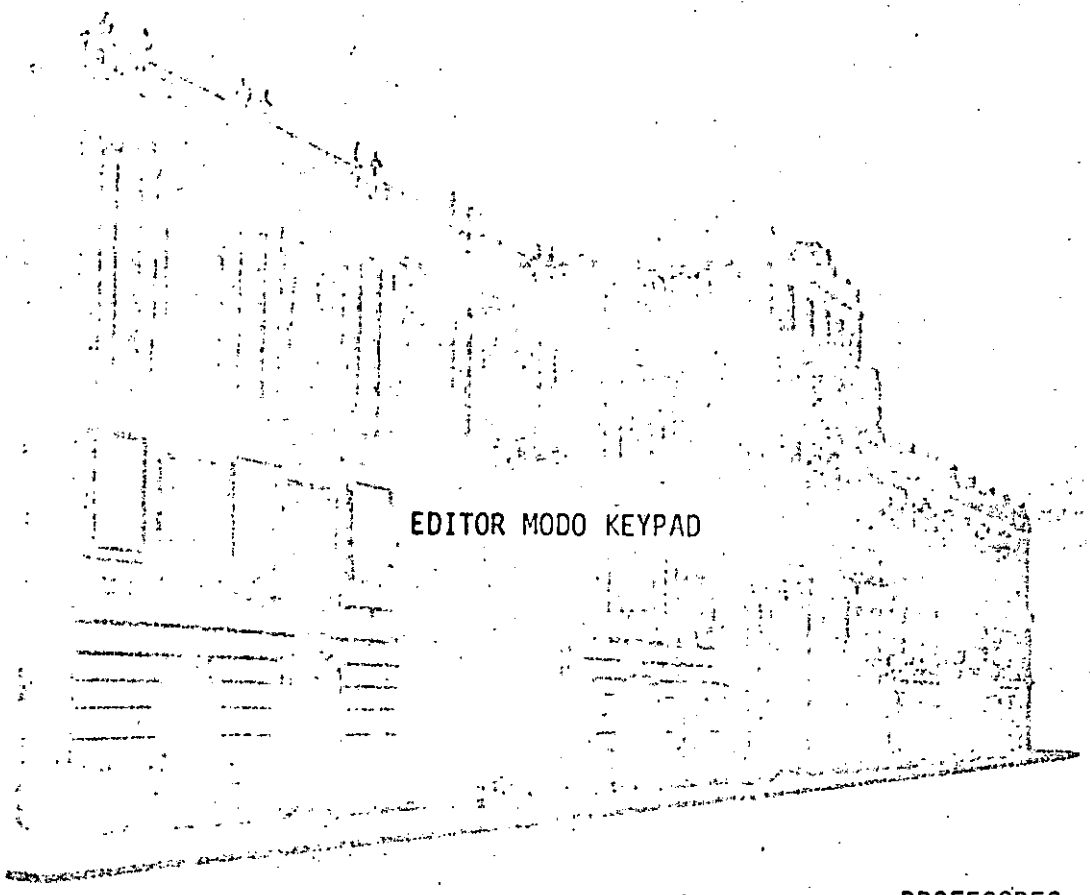
Si se omite el nombre del archivo, se salvará con el mismo nombre (el que se especificó al momento de entrar a edición).

Si se omite el rango, se escribirá todo el buffer.

Ejemplos:

*WRITE EDITOR.DAT DISK$CECAFI1:[OSITO]EDITOR.DAT;1 28 lines *

CURSOS: "INTRODUCCION AL SISTEMA VAX - 11/780" Y
"OPERACION Y ADMINISTRACION DEL SISTEMA VAX /VMS"
DEL 29 DE ABRIL AL 13 DE MAYO.
MEXICO, D.F.



PROFESORES:

ING. EDUARDO S. JALLATH CORIA.
ING. ALEJANDRO JIMENEZ GARCIA.
ING. SOCRATES A. MUÑIZ ZAFRA.
ING. HUMBERTO SANCHEZ SANDOVAL.

MAYO DE 1985.

CAPITULO 4

EDITOR MODO KEYPAD

4.1 TECLAS DE FUNCION ESPECIAL.

^ `' < > Las flechas sirven para moverse en toda la pantalla, en la dirección y sentido que señalen.

| PF1 | <número> Repite cualquier función del keyboard y del keypad, excepturando las siguientes: Specins, delete y los ctrl's.

|delete| La tecla DELETE sirve para borrar el último caracter escrito.

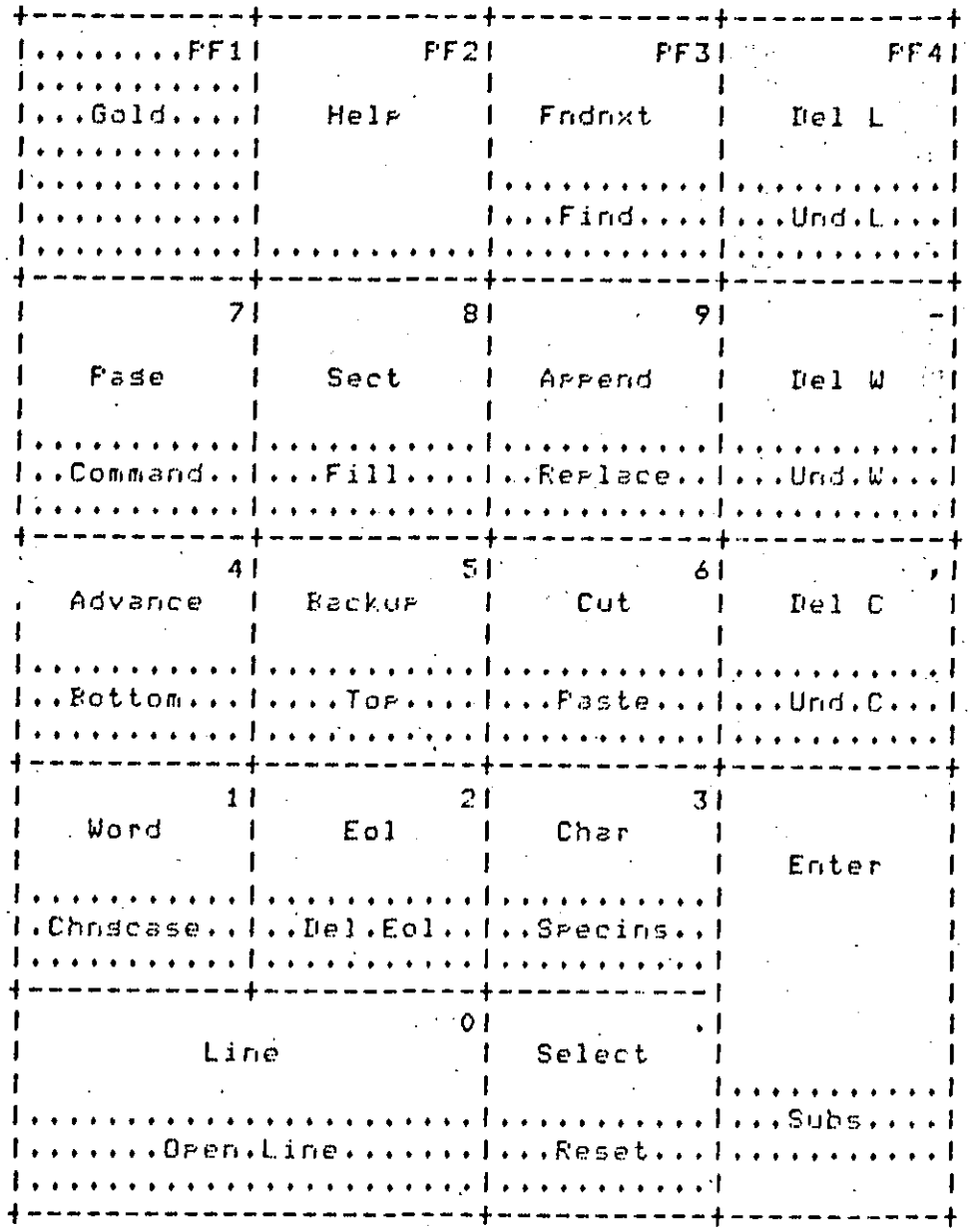
|BACK SPACE| La tecla BACK SPACE sirve para posicionar el cursor en la primera columna de la línea.

|LINE FEED| La tecla LINE FEED sirve para borrar la palabra inmediata anterior.

|CTRL| + |Z| La combinación simultáneo de las teclas control y Z hace que se regrese al editor modo línea

Los demás CONTROL ya fueron vistos en clases anteriores.

4.2 KEYPAD



El Editor EDT cuenta con varios buffers de trabajo los cuales son:

1. El buffer MAIN.

2. El buffer PASTE.

3. El buffer de un solo caracter.

4. El buffer de una sola palabra.

5. El buffer de una sola línea.

El buffer MAIN es donde se va a trabajar en la sesión de edición.

El buffer PASTE es temporal, ya que al salirnos de edición se destruye.

Las funciones de las teclas del KEYPAD son las siguientes:

4.3 GOLD

Varias de las teclas del KEYPAD tiene una doble función. Para determinar cada una de las funciones, nos ayudamos de la tecla PF1 (Gold). La tecla Gold sirve para activar el comando sombreado . Para hacerlo solo basta oprimir la tecla PF1 e inmediatamente el comando deseado.

4.4 HELP

El editor EDT modo KEYPAD cuenta con una explicación breve de cada uno de los comandos; para invocar esta ayuda basta con teclear PF2 (Help).

4.5 FIND

Las teclas PF1 PF3 (Gold Find) sirven para encontrar una cadena de caracteres dentro del buffer de trabajo, la busqueda la realizará en el sentido que tenga definido el cursor. (Ver Advance y Backup para definir el sentido del cursor.)

4.6 FNDNXT (FIND NEXT)

La tecla PF3 (Fndnxt) sirve para buscar la próxima ocurrencia de la cadena de caracteres seleccionada anteriormente. La busqueda la realizará en el sentido del cursor. (Ver Advance y Backup para definir el sentido del cursor.)

4.7 DEL L (DELETE LINE)

Si se desea borrar una línea, basta colocar el cursor al principio de la línea que se desea borrar y oprimir la tecla PF4 (Del L), esta línea se guardará en un buffer cuya capacidad es de una línea.

4.8 UND L (UNDELETE LINE)

Para recuperar la línea borrada con PF4 basta dar PF1 PF4 (Gold Und L) para que se haga una copia de la línea que se había borrado. Esta acción se puede repetir indefinidamente, con esto, queremos decir, que no se borra del buffer la línea al recuperarla.

4.9 PAGE

Cuando se tiene un archivo muy grande y se desea mover el cursor de página en página, basta con oprimir el 7 (Page) para colocarlo al principio de la página siguiente. El movimiento se hará en el sentido que se haya definido. (Ver Advance y Backup para definir el sentido del cursor.)

4.10 COMMAND

Cuando se desea dar un comando del editor modo línea, hay que oprimir las teclas PF1 7 (Gold Command), lo cual originará que aparezca el siguiente prompt: COMMAND, a continuación se podrá teclear cualquier comando del editor modo línea.

4.11 SECT (SECTION)

Cuando se oprime la tecla 8 (Sect) se avanzan 16 líneas en el buffer, el movimiento se realiza en el sentido del cursor. (Ver Advance y Backup para definir el sentido del cursor.)

4.12 FILL

Cuando se desea reformatear un texto, esto es, si se desea rellenar las líneas, se ocuparan las teclas PF1 8 (Gold Fill). Todas las palabras del rango seleccionado las acomodará en el menor número de líneas posible. (Ver Select para la definición del rango.)

4.13 APPEND

Para agregar texto al final del buffer auxiliar PASTE se usa la tecla 9 (Append) la cual moverá todo el rango seleccionado y lo pondrá al final del buffer PASTE. (Ver Select para la definición del rango)

4.14 REPLACE

Para cambiar un rango seleccionado, por el contenido del buffer PASTE basta teclear PF1 9 (Gold Replace). El buffer PASTE seguirá conservando el texto que tenía antes de teclear PF1 9. (Ver Select para la definición del rango.)

4.15 DEL W (DELETE WORD)

Cuando se desea borrar solamente una palabra, se oprime la tecla - (Del W), lo cual hará que se borre la palabra y se guarde en un buffer temporal cuya capacidad es de una palabra.

4.16 UND W (UNDELETE WORD)

Si se desea recuperar la palabra borrada con Del W, basta oprimir las teclas PF1 - (Gold Und W). La palabra del buffer de trabajo se conserva.

4.17 ADVANCE

Cuando se desea que el sentido de las operaciones sea hacia el final del buffers se teclea 4 (Advance), no importando el sentido que haya tenido, despues de teclear el sentido este quedará apuntando hacia el final del mismo.

4.18 BOTTOM

Cuando se desea mover el cursor hasta el fin del buffer oprimimos PF1 4 (Gold Bottom).

4.19 BACKUP

Cuando se desea que el sentido apunte hacia el principio del buffer se teclea PF1 4, al hacerlo el sentido quedará definido hacia el principio del mismo.

4.20 TOP

Para que el cursor se coloque al principio del buffer, basta con oprimir las teclas PF1 5 (Gold Top).

4.21 CUT

Para cambiar texto de un lugar a otro, primero hay que seleccionar un rango, posteriormente se oprime la tecla 6 (Cut) y enseguida se recupera en el lugar deseado. El CUT almacena en el buffer PASTE el rango seleccionado. (Ver select para la definición de un rango.)

4.22 PASTE

Para recuperar un texto que se encuentre en el buffer PASTE, basta oprimir las teclas PF1 6 (Gold Paste). El buffer PASTE seguirá conteniendo el texto.

4.23 DEL C (DELETE CHARACTER)

Al oprimir la tecla (Del C) se borrará el caracter que se encuentre en la posición del cursor. El caracter borrado se guardará en un buffer cuya capacidad es de un caracter.

4.24 UND C (UNDELETE CHARACTER)

Para recuperar un caracter borrado con el comando Del C basta oprimir las teclas PF1 , (Gold Und C). El caracter se seguirá conservando en el buffer de un caracter.

4.25 WORD

Con la tecla 1 (Word) se avanzará al comienzo de la siguiente palabra en el sentido del cursor. (Ver Advance y Backup para definir el sentido del cursor.)

4.26 CHNGCASE (CHANGE CASE)

Cuando se desea cambiar mayúsculas por minúsculas o viceversa hay que seguir el procedimiento mostrado a continuación:

1. Seleccionar el rango que se desee cambiar. Puede usarse para seleccionar el rango los siguientes comandos: Find y Select. Si se omite el rango asumirá el siguiente caracter

2. Después de seleccionado el rango se oprimen las teclas PF1 3 (Gold Chngcase) con lo que se cambiarán todas las letras minúsculas por mayúsculas y viceversa.

4.27 EOL (END OF LINE)

Para colocar el cursor al final de la línea hay que oprimir la tecla 2 (Eol).

4.28 DEL EOL (DELETE END OF LINE)

Si se oprimen las teclas PF1 2 (Del Eol) se borrará desde la posición actual del cursor hasta el fin de la línea. Esta información ya no se podrá recuperar.

4.29 CHAR

Si se oprime la tecla 3 (Char) el cursor avanzará al siguiente caracter. El movimiento lo hará en el sentido de movimiento del cursor. (Ver Advance y Backup para definir el sentido del cursor.)

4.30 SPECINS

Cuando se desea introducir al buffer de trabajo un caracter que no se encuentra en el keyboard, hay que seguir el procedimiento que se muestra.

1. Se oprime la tecla PF1 (Gold) y un número. El número será el valor ordinal del caracter ASCII que se desea introducir.

2. Finalmente, se oprimen las teclas PF1 3 (Gold Specins), con lo que el caracter ASCII correspondiente se colocará en donde se encuentre localizado el cursor.

4.31 LINE

Para avanzar al siguente principio de línea, basta con teclear el 0 (Line). El movimiento lo hará en el sentido de movimiento del cursor. (Ver Advance y Backup para definir el sentido del cursor)

4.32 OPEN LINE

Cuando se desea insertar una línea, hay que oprimir las teclas PF1 0. La línea insertada se colocará a la derecha del cursor.

4.33 SELECT

Cuando se desea seleccionar un rango, hay que seguir el siguiente procedimiento:

1. Oprimir la tecla . (select).

2. Mover el cursor a lo larso de todo el rango que se desee seleccionar. Para hacerlo se pueden usar los siguientes comandos o teclas: Back space, fndnxt, page, sect, bottom, top, word, eol, char o con el uso de las flechas. El rango seleccionado tendrá fondo obscuro.

4.34 RESET

Cuando se desea cancelar un Gold, se oprimen las teclas PF1 . (Gold Reset). También sirve para desactiver un rango.

4.35 ENTER

Para completar un comando se usa la tecla ENTER (Enter).

4.36 SUBS

Con las teclas PF1 ENTER (Gold Subs) se puede cambiar el contenido del buffer PASTE por una cadena de caracteres que se haya seleccionado. Para seleccionar la cadena de caracteres, primero se oprime la tecla . y posteriormente la cadena de caracteres.

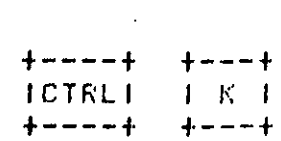
Cuando se desea cambiar varias ocuurencias, se sigue este procedimiento:

1. Se presiona el Select.
2. Se inserta el nuevo texto.
3. Se presiona Cut.
4. Se presiona el Find.
5. Oprimir ENTER cuando termine de meter el texto que se desea reemplazar.
6. Cada vez que se desee hacer una substitución, se oprimirá la tecla de Substitute.

4.37 COMO REDEFINIR LA FUNCION DE UNA TECLA.

Cuando se desea redefinir la función de alguna tecla, se usa el siguiente procedimiento:

1.



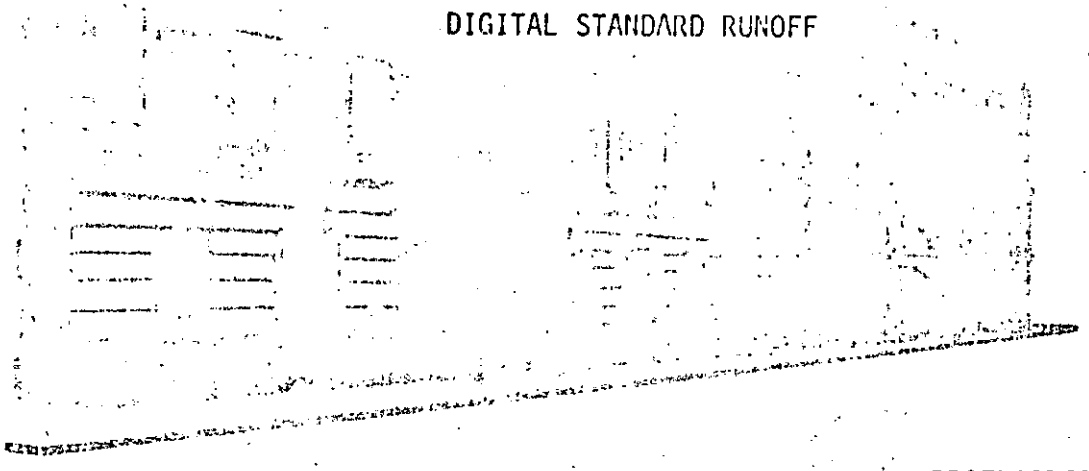
La combinación de las teclas de control y K

2. Despues de que aparezca el mensaje
Press the key you wish to define,
se oprime la tecla que se desea redefinir.

3. Hecho lo anterior aparecera lo siguiente
Now enter the definition terminated by enter,
enseguida del mensaje se escribirá la nueva definición y se terminará de definir con la tecla de ENTER. La definición es función
del teclado del KEYPAD.

CURSOS: "INTRODUCCION AL SISTEMA VAX - 11/780" Y
"OPERACION Y ADMINISTRACION DEL SISTEMA VAX /VMS"
DEL 29 DE ABRIL AL 13 DE MAYO.
MEXICO, D.F.

DIGITAL STANDARD RUNOFF



PROFESORES:

ING. EDUARDO S. JALLATH CORIA.
ING. ALEJANDRO JIMENEZ GARCIA.
ING. SOCRATES A. MUÑIZ ZAFRA.
ING. HUMBERTO SANCHEZ SANDOVAL.

MAYO DE 1985.

CAPITULO 5

DIGITAL STANDARD RUNOFF

5.1 INTRODUCCION

El propósito de estas notas, es el de describir el funcionamiento básico de DSR (Digital Standard Runoff), el cual es un Procesador de Palabras, llamado también, Formateador de Textos.

Cuando es utilizado DSR, se deben seguir tres pasos básicos :

a) Utilizar el EDITOR, para crear un archivo fuente
EDIT <nombre>.RNO

b) Utilizar DSR, para procesar el archivo y formatear el texto.
RUNOFF <nombre>.RNO

c) Imprimir el texto ya formateado
PRINT <nombre>.MEM

Para especificar un formato, desde DSR, se incluyen varios comandos en el archivo fuente; es posible indicar también,el tamaño de la página, cantidad de espaciamiento entre líneas, presentación de los elementos de una lista, etc.

Además de estos comandos, existen banderas (caracteres especiales), mediante los cuales podemos indicar subrayado de palabras, sobreimpresión ó empalme de caracteres (para especificar acentos) por ejemplo.

Los comandos han sido agrupados, de acuerdo con la función que realizan de la siguiente manera :

i. Formateo de las páginas

ii. Formateo del texto

iii. Definición de secciones (Encabezados, Capítulos, Apéndices)

iv. Banderas

Los comandos DSR son dados en el formato siguiente :

.<nombre comando>[.<nombre comando> ...] [;]

donde el primer punto debe ser colocado en la columna uno. El nombre del comando puede ser una abreviación del mismo.

De tal forma que los tres siguientes ejemplos, realizan la misma función

1. .FILL.JUSTIFY.NO KEEP; este es un ejemplo

2. .F.J.NK; este es un ejemplo

3. .FILL
.JUSTIFY
.NO KEEP
este es un ejemplo

5.2 FORMATEO DE PAGINA

5.2.1 Definición De Páginas

.PAGE SIZE [+/- n1][,+/- n2] --> .PS

Especifica el número máximo de líneas de texto en una página (n1), y el máximo número de caracteres por línea (n2).

(d) = .ps 58,60

.LAYOUT --> .LO

Las opciones son las siguientes :

.LAYOUT 0
.LAYOUT 1,n
.LAYOUT 2,n
.LAYOUT 3,n

donde "n", es el número de líneas en blanco, entre el fin de texto de la página, y el número de página de la parte inferior.

(d) = .LO 0

.LAYOUT 0 (default)

```
TITULO                          Page 2-2
SUBTITULO                       31 Oct 83
.
.
.
.
```

.LAYOUT 1,2

```
TITULO
SUBTITULO
.
.
.
2-2
```

.LAYOUT 2,2 (PAGINA PAR)

```
TITULO
SUBTITULO
.
.
.
.
2-2
```

.LAYOUT 2,2 (PAGINA NON)

```
                                TITULO
                                SUBTITULO
.
.
.
.
2-3
```

.LAYOUT 3,2

```
TITULO                          Page 2-2
SUBTITULO                       31 Oct 83
.
.
.
.
-35-
```

.FIRST TITLE --> .FT

Indica que el titulo, especificado con .TITLE, irá también en la primera página de un texto que no contenga capítulos ni apéndices.

.TITLE text --> .T

Especifica un título para ser encabezado en cada página, de

center.

acuerdo al valor de .LAYOUT .

.SUBTITLE text --> .ST

Especifica un subtitulo para ser encabezado, normalmente en la segunda línea, dependiendo del valor de .LAYOUT .

.DATE --> .D

Aparece la fecha del proceso en una de las líneas de encabezado en el formato 'dd mmm yy'.

.DISPLAY NUMBER y --> .DNM

Especifica la forma de numeración secuencial de páginas, de tal forma que 'y' puede tomar el valor de:

D --> decimales (d)

RU --> romanos mayúsculos

RL --> romanos minúsculos

RM --> solamente el primer romano mayúsculo

LU --> letras mayúsculas

LL --> letras minúsculas

LM --> solamente la primera letra mayúscula

.NUMBER PAGE [+/- n] --> .NMPG

Especifica un nuevo comienzo en la numeración secuencial de páginas.

5.3 FORMATEO DE TEXTO

5.3.1 Margenes, Rellenado Y Justificación

.LEFT MARGIN [+/- n] --> .LM

Especifica la posición del margen izquierdo. Por default n=0.

.RIGHT MARGIN [+/- n] --> .RM

Especifica la posición del margen derecho. Por default n=60

.FILL --> .F
.NO FILL --> .NF

Rellena cada línea con palabras de las subsecuentes líneas, mientras no se exceda del margen derecho. Si se utiliza .NO FILL, la parte final de cada línea es respetada.

.KEEP --> .K
.NO KEEP --> .NK

Respeta las líneas en blanco que tengamos en nuestro archivo de entrada.

.JUSTIFY --> .J
.NO JUSTIFY --> .NJ

Inserta en forma exacta los suficientes espacios entre las palabras para que todas las líneas terminen en el margen derecho.

.LITERAL --> .L
.END LITERAL --> .EL

Permite tener un texto formateado en la misma forma que en el texto original, inhabilitando todo comando dentro de este rango, y respetando únicamente el margen izquierdo y las banderas de Bolding (sobre impresión) y subrayado.

5.3.2 Espaciamiento Vertical

.BREAK --> .BR

Finaliza la línea en forma inmediata. La mayoría de los comandos realizan un break antes de ejecutarse el comando.

.BLANK n --> .B

Inserta exactamente el número de líneas en blanco que se especifiquen. Por default n=1.

.SKIP n --> .S

Inserta el número de líneas en blanco que se especifican multiplicadas por el valor que se le dé a .SPACING. Por default n=1.

.PAGE --> .PG

Comienza una nueva página. Para generar una serie de páginas en blanco se utilizan en forma alternada el .FIGURE y .PAGE.

.FIGURE [DEFERRED] n --> .FG[D]

Especifica un número de líneas en blanco, para posteriormente ser rellenadas al gusto del usuario. Utilizando 'DEFERRED' busca acomodar el espacio deseado en una misma página.

5.3.3 Espaciamiento Horizontal

.CENTER; text --> .CENTRE;text --> .C;text

Permite centrar el texto en una línea sencilla.

.INDENT [-]n --> .I

Causa que la primera línea en el texto siguiente sea sangrada "n" espacios a la derecha o izquierda de left margin.

5.3.4 Párrafos

.AUTOPARAGRAPH --> .AP

Si se comienza una línea con un espacio o con un <TAB>, automaticamente se formatea un párrafo según se especifique en .PARAGRAPH.

.PARAGRAPH n1, n2, n3 --> .P

Controla el espaciamiento y localización en la página para párrafos:

n1 --> sangrado de la primera línea del párrafo (def.=5).

n2 --> número de líneas en blanco que se deseen insertar antes de comenzar el párrafo, trabajando igual que .SKIP (def. = 1).

n3 --> número mínimo de líneas de un párrafo que debe estar en una misma página; si no hay espacio, comienza el párrafo en la siguiente página (def. = 2).

5.3.5 Listas

.DISPLAY ELEMENTS ['x',] y [,'z'] --> .DLE

Permite especificar la forma de numeración de los elementos de una lista.

x --> Caracter que precede al indicador del elemento (como un paréntesis izquierdo, por ejemplo). Por default es un blanco.
y --> Es el formato de numeración de los elementos. Ver .DISPLAY NUMBER.
z --> Caracter que sucede al indicador del elemento (como un paréntesis derecho, por ejemplo). Por default es un punto.

.LIST --> .LS
.END LIST --> .ELS

Especifica el comienzo de una lista de elementos. La orción termina con el .END LIST. Se pueden tener varias listas anidadas.

.LIST ELEMENT --> .LE

Especifica el comienzo de cada item en una lista.

5.4 DEFINICION DE SECCIONES

5.4.1 Apéndices Y Capítulos

.APPENDIX [texto] --> .AX

Especifica el comienzo de un apéndice, asignándole una letra identificadora y asoci'andole un nombre dado por el texto.

.CHAPTER [texto] --> .CH

Especifica el comienzo de un nuevo capítulo, asignándole un número decimal y asoci'andole un nombre dado por el texto.

5.4.2 Secciones

.HEADER LEVEL [+/- n] [title] --> .HL

Permite especificar el nivel de encabezado y el encabezado mismo.

Así :

```
.HL 1    -->   1.0
.HL      -->   2.0
.HL 2    -->   2.1
.HL      -->   2.2
.HL 3    -->   2.2.1
.HL 2    -->   2.3
```

5.5 BANDERAS

5.5.1 Uso Y Declaración

El uso de estas banderas puede habilitarse o dechabilitarse para su reconocimiento. Estas "banderas" no son más que caracteres especiales que nos ayudarán en la presentación del trabajo, pues mediante ellos podemos indicar que cierto caracter lleva acento ó implementar la letra "ñ"; indicar que cierta palabra debe de ser contenida en nuestro índice, ó simplemente una bandera que indique la acentación de un caracter especial, normalmente utilizado como bandera.

El formato utilizado para su declaración es :

.ENOIFLAGS <nombre bandera> --> .ENIFL

5.5.2 Utilidad

nombre	símbolo	Ejemplo
ACCEPT	_	Aster_*isco --> Aster*isco
BOLD	*	Wall*ace --> Wallace
		^*Irving* --> Irving*
COMMENT	!	Comen ! tario --> Comen
HYPHENATE	=	Ins=ti=tu=to --> Institu -
		to
Index	>	>Sabiduría (Aparecerá en el índice)
OVERSTRIKE	%	Ni~%ner'%ia --> Niñería
SUBSTITUTE	$$	$$DATE --> 11 Apr 84
		$$TIME --> 13:51:10
		$$YEAR --> 1984
		$$MONTH --> April
		$$DAY --> 11
		$$HOURS --> 13
		$$MINUTES--> 51
		$$SECONDS--> 10
SPACE	#	$$DATE#$$TIME --> 11 Apr 84 13:51:10
UNDERLINE	&	Letra &a --> Letra a
		^&S u b r a y a\& --> S u b r a y a

5.6 TABLAS DE CONTENIDO

Para indicar a la computadora que necesitamos una tabla de contenido, necesitamos :

1. Desde DCL (DIGITAL COMMAND LANGUAJE), se hace la asignación :
   toc :== $toc

2. Se crea un archivo <nombre>.MEM y uno más <nombre>.BTC
   RUNOFF/CONTENTS <nombre>.RNO

3. Se da la palabra TOC; nos preguntará por el <nombre> de nuestro archivo, tomando por default el tipo .BTC, además de otras seis preguntas acerca del formato de la tabla de contenido. Si a la primera pregunta contestamos con CTRL/z, asumirá las condiciones de default. Al finalizar, contestará con FINISHED. En este momento, se habrá creado un archivo <nombre>.RNT.

4. Se crea un archivo <nombre>.MEC
   RUNOFF <nombre>.RNT

5. Si no hubo errores durante la ejecución, se mandan imprimir los archivos; <nombre>.MEM, <nombre>.MEC.

5.7 CREACION DE INDICES

Desde el archivo fuente <nombre>.RNO deben ser especificadas las palabras que deseamos que sean incluídas en nuestro índice, mediante los comandos :

.INDEX topic[>subtopic1...>subtopic n] --> .X

que genera un número de página, ó

.ENTRY topic[subtopic1...>subtopic n] --> .Y

que no genera número de página.

Otra forma de hacer la indicación de que cierta palabra sea incluída en el índice, es mediante la bandera '>', declarándola previamente para tenerla habilitada con .FLAGS INDEX.

Para indicar a la computadora que necesitamos un índice, necesitamos :

1. Desde DCL (DIGITAL COMMAND LANGUAJE), se hace la asignación :
   tcx :== $tcx

2. Se crea un archivo <nombre>.MEM y uno más <nombre>.BIX
   RUNOFF/INDEX <nombre>.RNO

3. Se da la palabra TCX; nos presunterá por el <nombre> de nuestro archivo, tomando por default el tipo .BIX, además de otras seis presuntas acerca del formato del índice. Si a la primera presunta contestamos con CTRL/z, asumirá las condiciones de default. Al finalizar, contesterá con FINISHED. En este momento, se habrá creado un archivo <nombre>.RNX.

4. Se crea un archivo <nombre>.MEX
   RUNOFF <nombre>.RNX

5. Si no hubo errores durante la ejecución, se mandan imprimir los archivos, <nombre>.MEM, <nombre>.MEX.

Estas notas fueron elaboradas con el editor de textos RUNOFF.

Cualquier comentario para corregir, aumentar o modificar este texto favor de comunicarlo al Centro de Cálculo de la Facultad de Ingeniería.

GUIA DE ACCESO AL SISTEMA VAX 11-780

ENTRADA AL SISTEMA VAX11-780

1.1. QUE ES EL SISTEMA VAX 11/780 DEL CENTRO DE CALCULO.

La VAX-11/780 es una super minicomputadora de 32 bits por palabra de memoria, fabricada por Disital Equipment Corporation(DEC) , la cual forma parte de la serie de m'quinas m's populares utilizadas actualmente en las

INTRODUCCION:

ESTE MANUAL HA SIDO DESARROLLADO COMO UNA GUIA PARA EL USUARIO PRINCIPIANTE DE LA VAX/11-780 CON COMANDOS DCL Y TERMINALES TELEVIDEO. EN ESTA GUIA SE ENCONTRARAN LAS INSTRUCCIONES DE COMANDO MAS COMUNES ASI COMO SUS FORMATOS. MEDIANTE ESTA LO UNICO QUE SE PRETENDE ES DAR LA AYUDA NECESARIA PARA ENTRAR AL SISTEMA, ESCRIBIR, BORRAR, EDITAR, COMPILAR Y CORRER PROGRAMAS. TODO ESTO HECHO DE UNA FORMA SENCILLA Y CLARA QUE ESPERAMOS SEA DE UTILIDAD.

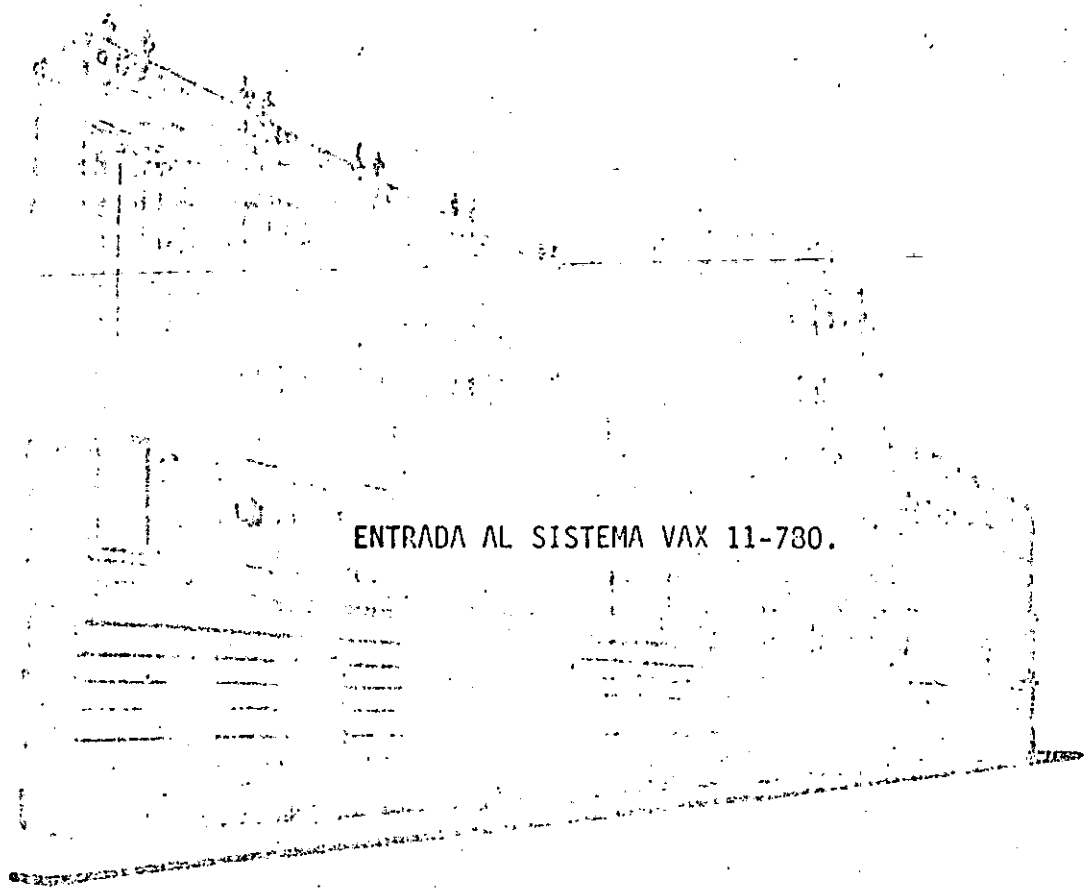
EDITH SILVA M.

JORGE ONTIVEROS

28 DE NOVIEMBRE DE 1983

INDICE

CURSOS: "INTRODUCCION AL SISTEMA VAX - 11/780" Y
"OPERACION Y ADMINISTRACION DEL SISTEMA VAX /VMS"
DEL 29 DE ABRIL AL 13 DE MAYO.
MEXICO, D.F.



ENTRADA AL SISTEMA VAX 11-780.

PROFESORES:
ING. EDUARDO S. JALLATH CORIA.
ING. ALEJANDRO JIMENEZ GARCIA.
ING. SOCRATES A. MUÑIZ ZAFRA.
ING. HUMBERTO SANCHEZ SANDOVAL.

MAYO DE 1985.

ENTRADA AL SISTEMA
VAX11-780

1.1. QUE ES EL SISTEMA VAX 11/780 DEL CENTRO DE CALCULO.

La VAX-11/780 es una super minicomputadora de 32 bits por palabra de memoria, fabricada por Digital Equipment Corporation(DEC) , la cual forma parte de la serie de máquinas más populares utilizadas actualmente en las Universidades. La capacidad actual es de 2 megabytes y se tiene posibilidades de crecimiento muy considerables. La VAX se basa en las micros y minis PDP también desarrolladas por DEC, de las cuales existen dos en la Facultad. La VAX cuenta con una buena cantidad de software a disposicion de los alumnos como son los compiladores BASIC, FORTRAN, COBOL, PASCAL; así como paquetes desarrollados por el personal del Centro, y una gran cantidad de rutinas de comandos de gran utilidad y fácil acceso.

El nombre VAX viene de 'Virtual Address eXtension' que significa extensión de direccionamiento virtual y esto viene porque la VAX tiene más capacidad de direccionamiento virtual que sus antecesoras las PDP y tiene capacidad de memoria virtual lo que le permite procesar programas más grandes que el tamaño de su memoria, gracias al concepto de paginación.

Para entender mejor esta primera parte, es necesario explicar que es una computadora.

Una computadora es una máquina con gran velocidad para realizar operaciones, con una gran capacidad de almacenamiento de información pero que necesita que le digamos que secuencia de pasos has que seguir.

Una computadora está compuesta por unidades de entrada, unidades de salida y la unidad central de procesamiento.

Las unidades de entrada pueden ser terminales, tarjetas, etc.

Las unidades de salida pueden ser impresoras, cintas, discos, terminales, etc.

La unidad central de proceso es el corazón de la

computadora y contiene la unidad aritmético-lógica, la unidad de control y la unidad de memoria. La unidad aritmético-lógica es la que realiza las operaciones aritméticas y booleanas, la unidad de control lleva el control de las operaciones (es la que gobierna) y la unidad de memoria es donde se almacenan los datos. La memoria puede ser memoria principal y memoria auxiliar (constituida por discos, cintas y/o diskettes).

La capacidad de las computadoras se mide en palabras o en bytes; una palabra es un conjunto de bytes (en Vax son dos bytes para WORD o cuatro bytes para LONGWORD. La VAX maneja registros de 32 bits (LONGWORD)), un Byte es un conjunto de bits (por lo general 8), y el bit es la unidad mínima de memoria, esto es, un lugar donde se puede almacenar un 0 ó un 1 (sistema binario de numeración).

Como la capacidad de las computadoras ha crecido mucho en los últimos años, se tienen factores que hacen manejables las cantidades de Bytes y son: Kilo=1024 (que es 2^10) entonces 1 Kb = 1024 bytes, Mega=1,048,144(que es 2^20) entonces 1 Mb = 1,048,144 bytes.

El sistema Operativo es el VMS (Virtual Memory System) que significa Sistema de Memoria Virtual el cual permite un manejo eficiente de la memoria física gracias al principio de la Memoria Virtual. El Sistema Operativo es el administrador de los recursos de la computadora.

Una página de memoria en Vax está constituida por 512 bytes que es a su vez un bloque en disco.

La definición de computadora que se proporciona es muy simple, para poder diferenciar computadoras, se subdividen las Unidades, por lo que la descripción de la configuración se realiza en forma más detallada.

La configuración actual del sistema es:

Subsistema de Consola:
- 1 microprocesador LSI-11
- 1 unidad de diskettes de 256 Kb
- 1 terminal LA-120

Unidad Central de Proceso con:
- 1 Respaldo de baterias para salvar a disco los procesos.
- 1 Acelerador de punto flotante.
- 1 Memoria caché.

Synchronos Backplane Interface
que es un canal que comunica a la memoria, al Massbus y al Unibus con el Procesador, con una capacidad de 13.3 Mb/seg.

Memoria principal:	1	controlador de memoria con 8 módulos de 256 Kb c/u = 2 Mb.
Massbus de 2 Mb/seg con:		
Almacenamiento Secundario	3	unidades de disco de 176 Mb cada una.
Almacenamiento en cinta	1	unidad de cinta magnética de 800/1600 bpi y 45 ips
Unibus de 1,5 Mb/seg con:		
Impresión	1	impresora de 600 lpm
	1	impresora graficadora
Lectura	1	lectora de tarjetas de 300 tarjetas por minuto
disco flexible	1	unidad dual de diskkette de 256 kb c/u
terminales	3	multiplexores con capacidad de 8 terminales cada uno.
	20	terminales de video VT 100
	4	terminales de papel LA 120 de 120 caracteres por seg. cada una.

Antes de entrar en sesión (conectarse con el sistema para utilizar los recursos de la computadora) es necesario describir el funcionamiento de algunas teclas importantes.

El teclado de las terminales se parece al de una máquina de escribir normal, pero tiene una serie de teclas adicionales como son:

RETURN que se utiliza para finalizar la información que le estamos dando a la computadora.
DELETE que sirve para borrar el caracter que acabamos de teclear.
CAPS LOCK que sirve para escribir todo con letras mayúsculas pero no para escribir los caracteres que están colocados en la parte superior de las teclas con dos caracteres.
SHIFT sirve para escribir los caracteres alternos de las teclas con dos caracteres(ejemplo 4 y $ están en la misma tecla).
CTRL sirve para dar instrucciones especiales para la computadora como seria CRTL U (oprimir simultaneamente las teclas CTRL y U) que nos permite desechar una linea completa.
NO SCROLL sirve para detener/continuar con el desplegado en la pantalla.

ESC que permite proporcionar una secuencia de ESCape para modificar las características de la pantalla o auxiliarnos en el manejo de la misma.
BACK SPACE genera un caracter (CTRL/H) que hace retroceder un caracter el cursor pero no borra el caracter encimado (aunque en la pantalla así lo parezca).
LINE FEED baja una línea en la pantalla (en modo local).

1.2. USERNAME.

Llegando a la sala de terminales lo primero que se tiene que hacer es encender el switch interruptor que se encuentra del lado izquierdo inferior posterior de la terminal. Tardará un breve lapso en aparecer un cursor, este tiempo que tarda la terminal es el que utiliza para calentarse, ya que la pantalla funciona como un simple cinescopio de television. Ya que apareció el cursor (un cuadro) hay que oprimir la tecla <return>, con la que se llama la atención del procesador y que pedirá el USERNAME asignado. Se deberá teclearlo y oprimir <return>

1.3. PASSWORD.

El PASSWORD es una clave secreta que va asociada con el USERNAME. En este momento el sistema está pidiendo esa clave, deberá escribirla pero tome en cuenta que no se desplegará en pantalla, sino que será interpretado por la máquina únicamente. Después de realizar esta operación oprimir <return> y se estará ya dentro del sistema. La operación general es:

password: No se despliega en pantalla!!!

1.4. MENSAJES DEL SISTEMA.

Lo primero que se verá al entrar al sistema será un mensaje que dirá Bienvenido al Sistema VAX/VMS V3.3 (Cecafi) posteriormente se podrán desplegar mensajes del sistema al usuario, tales como advertencias, mensajes varios, etc... Aparecerá el PROMPT (indicador), el cual es un $ (que indica que estás en el DCL ó Disital Command Language).

Cuando esto sucede se dice que el usuario ha entrado en sesión lo que significa que se puede hacer uso de los recursos de la computadora.

1.5. DIFICULTADES DE ACCESO.

Los mensajes que podrá dar la máquina en caso de una dificultad de acceso son los siguientes:
a) Que aparezca un mensaje que diga USER AUTORIZATION FAILURE , esto es que se cometió un error al poner el USERNAME o el PASSWORD por lo que tendrá que iniciar nuevamente el proceso de entrada.
b) Si la computadora no responde puede ser que en ese momento el sistema esté fuera de servicio momentaneamente ('caido') por lo que deberá esperar el mensaje correspondiente al reinicio de operaciones.
c) Si se entra en sesión pero no permite ver el directorio, ni los archivos puede ser que el disco al que pertenece la clave no esté disponible, por lo que deberá terminar la sesión y preguntar a que hora estará disponible el disco.
En caso de cualquier problema con el equipo lo conveniente será consultar a los encargados en el mostrador de reservaciones o al asesor que se encuentre en ese momento en las terminales.

1.6. MANEJO DE PANTALLA

La terminal VT100 de la computadora tiene ciertas características que se pueden modificar para una sesión y otras más que se pueden manejar desde programa.

En la parte superior del teclado aparecen unos LEDs y una serie de palabras en inglés en letras pequeñas que son:

1) SET/CLEAR TAB que nos va a permitir borrar o fijar tabuladores de la pantalla.
2) CLEAR ALL TABS que borra todos los tabuladores.
3) LINE/LOCAL que nos permite pasar de modo LINEA a modo LOCAL y viceversa.
4) SETUP A/B nos permite pasar del SET UP A al SET UP B y viceversa.
5) TOGGLE 1/0 nos va a permitir modificar las características de la pantalla.
6) TRANSMIT SPEED permitirá modificar la velocidad de transmisión de la terminal.
7) RECEIVE SPEED permitirá modificar la velocidad de recepción de la terminal.
8) 80/132 COLUMNS nos permite cambiar de 80 a 132 columnas y viceversa.
9) RESET ocasiona que todos los cambios que se hayan realizado se desechen y la terminal tome las características de Default.

Para poder hacer estos cambios es necesario oprimir la tecla de SET-UP, y se entra al SET UP A donde se puede ver la posición de los tabuladores.

Para pasar al SET UP B se oprime la tecla 5 donde se observa las velocidades de recepcion y transmision (que deben estar a 9600 en las terminales de Edificio Principal), y 4 conjuntos de 4 bits que determinan las características de la terminal. Para ver su significado se recomienda levantar con mucho cuidado el teclado y ver la parte inferior del mismo.

El significado de cada uno de estos es:

BYTE 1

SCROLL 0=RAPIDO 1=SUAVE
controla la velocidad de desplegado en la pantalla
AUTOREPEAT 0=APAGADO 1=ENCENDIDO
permite la repetición de caracteres al oprimir la tecla respectiva una sola vez.
SCREEN 0=FONDO NEGRO 1=FONDO BLANCO
Cambia el fondo de la pantalla
CURSOR 0=SUBGUION 1=CUADRO
Cambia el caracter del cursor

BYTE 2

MARGIN BELL 0=APAGADO 1=ENCENDIDO
Hace sonar la campana cuando se llega a la columna 72'
KEYCLICK 0=NO SUENA 1=SUENA
Permite que la campana suene cada vez que se oprime una tecla. (pero no igual de intenso que el MARGIN)
ANSI/VT52 0=VT52 1=ANSI
Nos permite simular una terminal VT 52 ...
AUTO XON XOFF 0=APAGADO 1=ENCENDIDO
Señales de control para generar códigos síncronos

BYTE 3

0=# 1=£ Libra Esterlina
Hace que la terminal despliegue un # o un caracter de Libra Esterlina(£)
WRAP AROUND 0=NO BAJA 1=SI BAJA
Cuando se llega a la columna 80 el cursor se baja a la siguiente línea.
NEW LINE 0=NO BAJA 1=SI BAJA
Cuando se oprime RETURN el cursor regresa a la columna uno y baja de renglón.
INTERLACE 0=APAGADO 1=ENCENDIDO
Se utiliza para gráficas de alta resolución.

BYTE 4

PARITY SENSE 0=NON 1=PAR
Determina que tipo de paridad se va a verificar
PARITY 0=APAGADA 1=ENCENDIDA
Indica si se verifica la paridad
BITS PER CHAR 0=7 (ASCII) 1=8 (EBCDIC)
Nos permite seleccionar el código de transmisión de la información. La VAX trabaja en ASCII.
POWER 0=60 HZ 1=50 HZ
Indica la frecuencia de la energía eléctrica

Para modificar alguna de las características es necesario avanzar con las flechas que están en el teclado hasta estar arriba del bit que queremos modificar y entonces modificarlo con la tecla 6.

Si se desea hacer pruebas del SET UP, se recomienda hacerlas cambiando primero a modo local (estando en SET UP A) y que al final de la sesión le des RESET (SET UP A con 0)

Existen además secuencias de teclas que nos permiten el manejo de la pantalla desde un programa. Para ver que hacen se recomienda trabajar en modo LOCAL antes de empezar a programarlas. Las más comunes son:

ESC[Pn A subo el cursor n líneas
ESC[Pn B baja el cursor n líneas
ESC[Pn C avanza el cursor n posiciones
ESC[Pn D retrocede el cursor n posiciones
ESC[Pl;Pc H mueve el cursor a la línea y columna indicadas por los dos parámetros.
ESC[Pl;Pc f mueve el cursor a la posición indicada por los dos parámetros
ESC#3 agranda la mitad superior de las letras que se encuentren en esa línea
ESC#4 agranda la mitad inferior de las letras que se encuentren en esa línea
ESC#5 regresa la línea al tamaño normal
ESC#6 agranda a doble ancho las letras que se encuentren en esa línea
ESC[Ps J borra caracteres en función de s :
0 = borra desde la posición actual hasta el final de la pantalla
1 = borra del inicio de la pántalla hasta la posición actual
2 = borra toda la pantalla pero el cursor no se mueve de su posición.

ESC[PsK borra caracteres en la línea para s :
0 = borra desde la posición hasta el final
1 = borra del inicio de la línea hasta la posición el cursor
2 = borra toda la línea

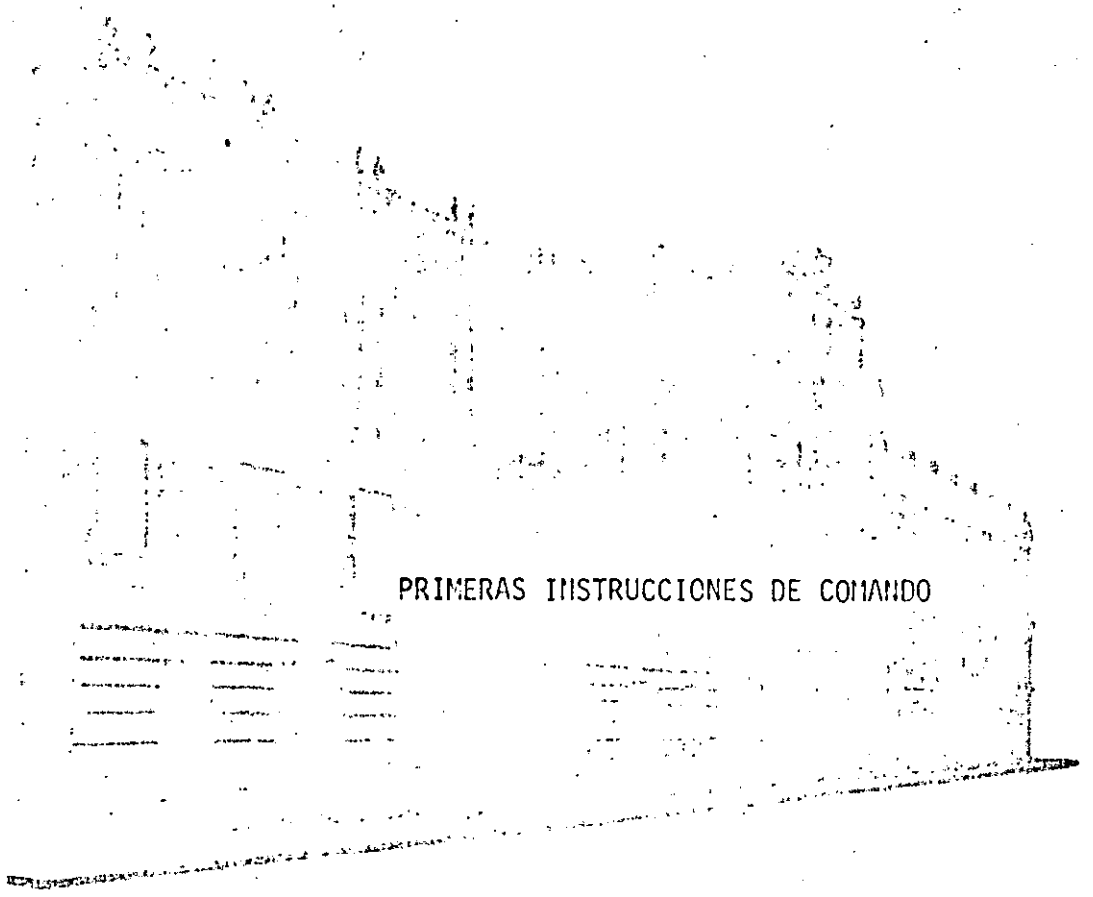
ESC H fija el tabulador donde está el cursor

ESC[Psg borra los tabuladores, si s = 0 borra el tabulador en la posición del cursor y si s=3 borra todos los tabuladores.

ESC[Ps;...;Psm permite modificar parte de la pantalla en función de s :
0 = atributos apagados
1 = incrementa la intensidad
4 = subraya
5 = parpadea
7 = con fondo blanco

ESC[Pt;Pb r define la región del 'scroll' con t=línea superior y b=línea inferior.

CURSOS: "INTRODUCCION AL SISTEMA VAX - 11/780" Y
"OPERACION Y ADMINISTRACION DEL SISTEMA VAX /VMS"
DEL 29 DE ABRIL AL 13 DE MAYO.
MEXICO, D.F.



PRIMERAS INSTRUCCIONES DE COMANDO

PROFESORES:

ING. EDUARDO S. JALLATH CORIA.
ING. ALEJANDRO JIMENEZ GARCIA.
ING. HUMBERTO SANCHEZ SANDOVAL.
ING. SOCRATES A. MUÑIZ ZAFRA.

MAYO DE 1985.

2
PRIMERAS INSTRUCCIONES DE COMANDO

EL SISTEMA VAX 11-780 UTILIZA UN SIMBOLO PARA ESPECIFICAR SUS DISTINTOS MODOS DE OPERACION, ENTRE ESTOS SE TIENE:

COMANDOS DEL SISTEMA Caracterizada por un signo de pesos a la izquierda.

$

COMANDOS DEL EDITOR Caracterizada por un asterisco a la izquierda.

*

DENTRO DEL EDITOR Caracterizada por el mensaje de [EOB] en la parte inferior de la pantalla.

[EOB]

2.1. Dentro de las instrucciones básicas de operación del sistema VAX debemos considerar DIR y EDT. La instrucción DIR es la forma abreviada de DIRECTORY. DIR es una de las instruciones más frecuentes que se utilizará al estar operando el sistema. Este DIRECTORIO muestra el número de de programas y archivos que se tienen, sus nombres, características y versiones. Cada nombre de archivo se estructura de la siguiente manera.

DISPOSITIVO:[DIRECTORIO]NOMBRE-DE-ARCHIVO.EXTENSION;VERSION

El dispositivo es el nombre donde está físicamente guardado el archivo esto es disco, cinta, diskette, tarjetas; este nombre se forma con DB para disco, LP para impresora, MT para cinta magnética, DY para diskette, CR para la lectora, y va seguido del nombre del controlador que consta de una letra (A,B,C, etc.) y el número que ocupa dentro del controlador (0,1,2,3, etc.) por lo que para la configuración actual de la VAX los nombres son:

DBA0: para el disco cero
DBA1: para el disco uno
DBA2: para el disco dos
LPA0: para la impresora
LPB0: para la impresora graficadora
CRA0: para la lectora de tarjetas
DYA0: para el dispositivo de diskettes
MTA0: para la cinta magnética
TTA0: a TTA7:, TTB0: a TTB7:, y TTC0: a TTC7: para las terminales

El nombre del directorio es el mismo que el del USERNAME pero puede ser también el de algún subdirectorio (posteriormente se verá qué es y como se definen éstos)

El nombre del archivo lo asigna el usuario mediante un nombre que indique la finalidad de dicho archivo, este puede ser de 1 a 9 letras y números sin incluir caracteres especiales. La extensión(tipo) también se la asigna el usuario cuando se piensa escribir un programa en algún lenguaje en particular, ya sea BASIC, FORTRAN, PASCAL o DATOS. El usuario deberá asignar la extensión con las primeras tres letras del lenguaje que se esté utilizando esto es; suponsamos que queremos dar el nombre a un programa, a este se le va a llamar EJEMPLO y se escribirá en varios lenguajes:

```
BASIC        EJEMPLO.BAS
FORTRAN      EJEMPLO.FOR
COBOL        EJEMPLO.COB
PASCAL       EJEMPLO.PAS
```

Ejemplos de nombres de archivos:

```
DBA1:[CLASE]EJEMPLO.BAS;3
MTA0:[CINTA]TRABAJO.FOR;6
DBA2:[JUANITO]DATOS.PAS;4
```

CONSIDERACIONES:

Si no se proporciona el dispositivo, la computadora asume que se trata del dispositivo donde está asignada la clave; si no se proporciona el directorio se asume que es el de la clave; y si no se proporciona la versión, la computadora asume que queremos la última versión del archivo. Con todo lo anterior los nombres se pueden simplificar a:

```
EJEMPLO.BAS
DATOS.PAS
```

Para saber que archivos se tienen en el directorio o subdirectorio, existe el comando DIRECTORY que nos proporcionará la lista o el mensaje correspondiente.

Su forma abreviada es $ DIR <return>

Otras dos instrucciones útiles son CREATE DIR y SET DEF.

La primera permite crear subdirectorios dentro del espacio en disco. Esto es recomendable cuando la clave es compartida entre dos usuarios y/o se va a tener archivos con programas de diferentes asignaturas. El CREATE DIR se debe usar una sola vez.

Su formato general es:
$ CREATE/DIRECTORY especificación del directorio
donde especificación es [directorio.subdirectorio]
Ejemplo:
$ CREATE/DIR [CURSOS.EGK] <return>

Se crea el subdirectorio [.EGK] (iniciales de Ernesto Gutierrez Kuri) en el directorio [CURSOS]
$ CREATE/DIR [AMH200.BASIC]
Se crea el subdirectorio BASIC en la clave(directorio) [AMH200].

Para poder trasladarnos a éstos subdirectorios debemos emplear el comando SET DEFAULT.

$ SET DEFAULT [directorio.subdirectorio]
$ SET DEF [CURSOS.EGK] <return>
$ SET DEF [.BASIC] <return>

En el primer ejemplo del directorio [CURSOS] se va al subdirectorio [.EGK], y en el segundo ejemplo se le indica de donde se está, se quiere ir al subdirectorio [.BASIC], en este caso si no se escribe el nombre del directorio, asume que es en el directorio en que se encuentra actualmente. Mientras no se este muy familiarizado con el uso del equipo es conveniente utilizar la forma completa.

Para regresar al directorio principal basta escribir:
$ SET DEF [CURSOS] si nuestra clave es CURSOS o
$ SET DEF [AMH200] si nuestra clave es AMH200, etc.

Para saber en que subdirectorio se encuentra el usuario, sobre todo cuando se tienen varios, será necesario utilizar la instrucción

$ SHOW DEFAULT que lo proporcionará.

Asociados al comando SHOW hay muchos parámetros, entre los cuales están:

$ SHOW TIME que proporciona la fecha y la hora.

$ SHOW QUOTA que dá el UIC, cuantos bloques se han utilizado, cuantos quedan disponibles y en que disco se está trabajando.

$ SHOW DAYTIME es equivalente a $ SHOW TIME.

2.2. COMANDO EDT

Se utiliza cuando se quiere crear algún programa o archivo, por lo que se deberáescribir el comando EDT cuando se esté en modo comando del sistema, esto es que la maquina genere un signo de $ a la izquierda.
$ Se escribe EDT...

```
$ EDT <return>          <return> IMPLICA PRESIONAR LA TECLA
                                 RETURN ! ! !
```

Y aparece en el display... $-file:
En este momento se tienes la capacidad de dar el nombre y la extension al programa, lo cual se deberá hacer y presionar <return>.

En ese momento aparece el letrero:

```
Input file does no exist [EOB]
*
```

En este momento se deja de estar en DCL para estar en Editor; en el MODO EDITOR se puede crear y/o modificar archivos mediante los comandos propios del Editor que son:

CHANGE	CLEAR	COPY	DEFINE
DELETE	EXIT	FILL	
FIND	HELP	INCLUDE	INSERT
JOURNAL	KEYPAD	MOVE	
PRINT	QUIT	RANGE	REPLACE
RESEQUENCE	SET	SHOW	
SUBSTITUTE	TABS	TYPE	WRITE

Uno de los comandos más importantes es el HELP que ayuda a recordar el formato de los comandos, hay que entrar al Editor para ver su uso.

```
$ EDT AA. <return>

Input file does not exist
[EOB]

* HELP  <return>
HELP

You can get help on a topic by typing

     HELP topic subtopic subsubtopic...
```

```
A topic can have one of the following forms:

     1. An alphanumeric string
          (e.g. a command name, option, etc.)
     2. The match-all or wild card symbol (*)

     Examples:        HELP SUBSTITUTE NEXT
                      HELP CHANGE SUBCOMMAND
                      HELP CH

If a topic is abbreviated, the text for all topics
which match the abbreviation is displayed.

Additional information available:

CHANGE      CLEAR       COPY        DEFINE      DELETE
EXIT        FILL        FIND        HELP        INCLUDE
INSERT      JOURNAL     KEYPAD      MOVE        PRINT
QUIT        RANGE       REPLACE     RESEQUENCE  SET
SHOW        SUBSTITUTE  TABS        TYPE        WRITE


* EXIT

$ DBA2:[JORGEJAA.;1 No lines

  Los comandos más utilizados son:
* INSERT
Su formato general es:
* INSERT [rango] [; linea a ser insertada]
Donde rango puede ser:
 1) un número de línea
 2) la palabra BEGIN que indica el inicio del archivo
 3) la palabra END que indica el fin del archivo
 4) cualquier combinación de éstas separadas por dos puntos
    (:) o la palabra THRU.
 5) la palabra WHOLE que significa todo
 6) Rango + número
 7) Rango - número
 8) 'string' a buscar
 9) LAST
10) .  ( linea actual)
11) Número de linea  cuantas más
12) BEFORE
13) REST

Todo lo que esté encerrado entre [] significa que es
opcional, si no se especifica rango se asume la línea
actual y si no se especifica línea a ser insertada se asume
que es un conjunto de ellas que se darán a continuación.
```

Ejemplos:
* I 8 agrega líneas antes de la línea numerada con 8
* I 8; B = 2 * A
inserta la línea que aparece en el comando antes de la 8
Para salirse de INSERT se oprimen simultaneamente CRTL y Z
* INSERT
Permite introducir el archivo (programa), el cursor inicia en la columna 10 aproximadamente pero es la primera columna del archivo. Cada vez que damos <return> el cursor pasa a la siguiente línea. Para terminar hay que utilizar CTRL/Z.

* SUBSTITUTE
Permite corregir el archivo.
Su forma general es:
* SUBSTITUTE/string1/string2/[rango][BRIEF:n][/QUERY]
donde string1 es lo que está incorrecto, string2 es como se desea que quede, rango se explicó en el comando anterior, BRIEF:n permite indicar que no despliegue toda la línea sino unicamente 'n' caracteres de ésta y el calificador QUERY le indica a la computadora que se quiere confirmar antes de hacer cada substitución y las posibles respuestas son: Y(si), N(no), A(todas las restantes), y Q(que ya no efectúe ningún cambio más).

Ejemplos:
* S/heror/error/5
En este caso solo la línea 5 se modifica
* S/alumno/estudiante/WHOLE/QUERY
Se desea que efectúe las substituciones en todo el archivo pero que vaya preguntado antes de hacerlas.

* REPLACE
Permite substituir líneas existentes por las que se van a teclear a continuación; al terminar hay que dar CTRL/Z.
* REPLACE [rango][;línea a ser insertada]

* MOVE
Mueve líneas de texto de un lugar a otro.
* MOVE [rango1] TO [rango2] [/QUERY]
donde rango1 es el conjunto de líneas que se quieren mover y rango2 es el lugar donde se quieren que estén.

* COPY
Esta instrucción permite duplicar un conjunto de líneas en otro lugar del archivo.
* COPY [rango1] TO [rango2] [/QUERY] [/DUPLICATE:n]
Donde n es el número de veces que se desea duplicar el texto.

* RESEQUENCE
Cuando se desea que la numeración interna del archivo se normalice o modifique, se puede usar ésta instrucción.
* RESEQUENCE [rango] [/SEQUENCE] [:inicio [:incremento]]
* RES 2:5/SEQ :2:3
Se está pidiendo que el bloque de líneas que va del 2 al 5 se renumere iniciando en 2 y con incrementos de 3 en 3. Se hace notar que entre 2 y 5 pueden existir una gran cantidad de líneas.

* TYPE
Permite ver en la terminal el archivo, con la numeración interna proporcionada por el Editor.
* TYPE [rango] [/BRIEF:n] [/STAY]
donde el calificador /STAY ocasiona que el apuntador de línea no avance de su posición actual.
* TYPE 'string' busca la ocurrencia del string y nos despliega la línea; si se añade ALL desplegará todas las líneas que contengan el string

* FIND
Este comando permite ubicarse en una línea, esto es, el apuntador de líneas se mueve a ésta.
* FIND [rango]

* DELETE
Permite borrar líneas del archivo
* DELETE [rango]

* EXIT
Permite salirse del Editor salvando el archivo en el disco.

* QUIT
Se sale del Editor pero se desecha todo lo realizado en la sesión de Edición.

* INCLUDE
Con este comando se puede traer un archivo en el disco e incluirlo en el archivo que se está editando.
* INCLUDE nombre-de-archivo [rango]

* PRINT Manda al espacio en disco una copia del archivo que se está editando, pero incluye la numeración interna.
* PRINT nombre-de-archivo [rango]

* WRITE
Crea un archivo en nuestro espacio en disco, sirve para crear protecciones temporales cuando el sistema puede tener fallas continuas y nuestros archivos son muy grandes.
* WRITE nombre-de-archivo [rango] [/SEQ [:inicio[:inc.]]]

* CHANGE Este comando permite cambiarse al otro modo del Editor pero que consume más recursos de computadora.

Estaremos en modo INSERT.

*C <return>

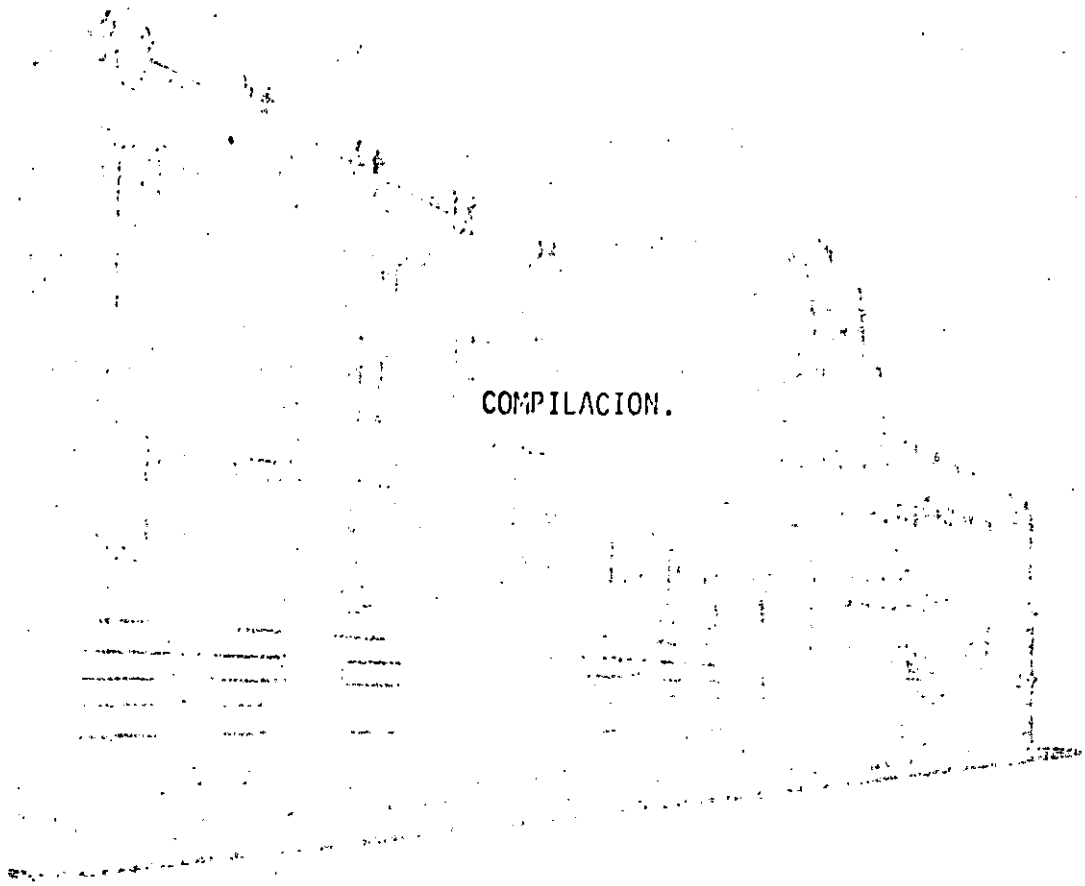
Esto deja "una hoja en blanco" donde se podrán escribir programas y archivos. Dejará en la pantalla un mensaje que dice [EOB] y en ese momento, el usuario estará dentro del EDITOR de caracteres, habiendose dejado el modo línea del COMANDO EDITOR. Este mensaje de [EOB] indica la dimensión del archivo en el que se está operando. Cuando se esté escribiendo un programa el usuario deberá tomar en cuenta los formatos de los diferentes lenguajes. Cuando se termine de escribir el programa se dará el comando de salida del modo CARACTER al modo LINEA, mediante el control CTRL-Z oprimiéndolas al mismo tiempo, con lo que se esté ya en modo LINEA. Para poder salir a DCL(comando del sistema) se tienen dos opciones principales que son:

QUIT o EXIT esto es:

*QUIT <return> o
*EXIT <return>

La instrucción QUIT borra lo que se haya escrito, si es la primera vez que se llama por EDT al programa, en caso contrario deja la versión anterior, y el comando EXIT graba el archivo o programa en el area de trabajo por lo que se podrá contar con el en el directorio.

CURSOS: "INTRODUCCION AL SISTEMA VAX - 11/780" Y
"OPERACION Y ADMINISTRACION DEL SISTEMA VAX /VMS"
DEL 29 DE ABRIL AL 13 DE MAYO
MEXICO, D.F.



COMPILACION.

PROFESORES:

ING. EDUARDO S. JALLATH CORIA.
ING. ALEJANDRO JIMENEZ GARCIA.
ING. SOCRATES A. MUÑIZ ZAFRA.
ING. HUMBERTO SANCHEZ SANDOVAL.

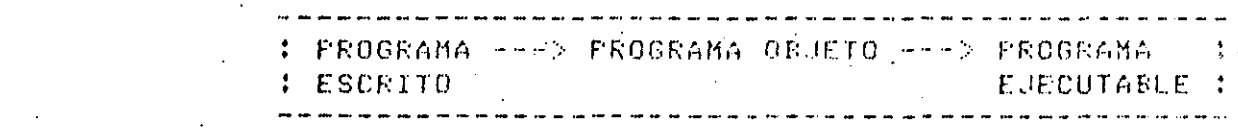
MAYO DE 1985.

3
COMPILACION

3.1. INTRODUCCION A LA COMPILACION

La compilación es el procedimiento mediante el cual la computadora cambia los programas que se crearon mediante el EDITOR, a lenguaje máquina o binario, realizando la detección de errores durante este proceso. Este procedimiento, cuando se realiza en VAX consta de dos partes diferentes, la primera genera las extensiones OBJ u objeto que son la traducción de los programas realizada por el compilador, en esta parte es donde se identifican los errores de programación que son generalmente mala sintaxis de los verbos del programa o instrucciones inválidas. Suponiendo que el programa no tuvo errores, el compilador, el cual, es el traductor, realiza la traduccion perfectamente; entonces se debe generar, mediante la instrucción LINK, la extensión EXE del programa; en este momento tu programe esta listo para correr. Lo que hace el LINK es ligar el módulo objeto con otros módulos objetos necesarios para la corrida, éstos módulos pueden ser creados por los usuarios o por el sistema.

Este procedimiento es:

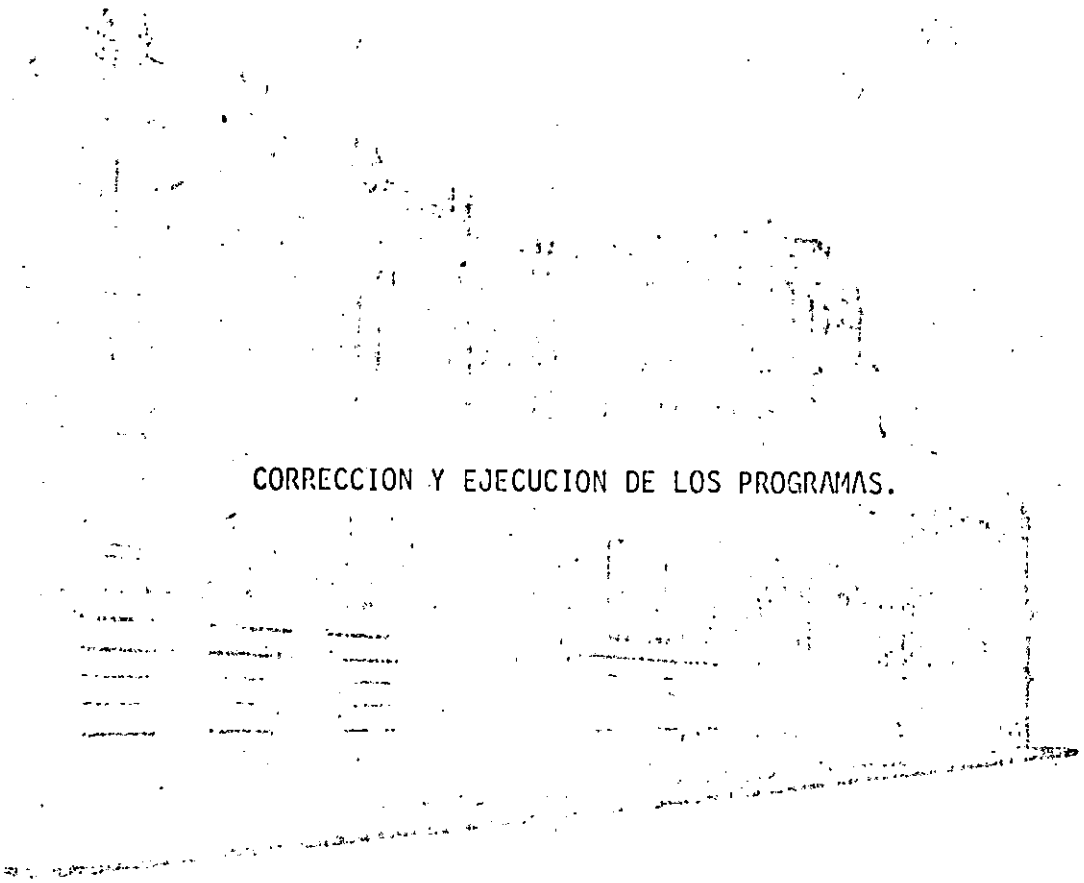


3.2. PASOS A SEGUIR SEGUN LENGUAJE DE PROGRAMACION:

PROGRAMA ESCRITO EN	PROGRAMA OBJETO	PROGRAMA EJECUTABLE
BASIC	$ BAS nombre-pros <ret>	$ LINK nombre-pros <ret>
FORTRAN	$ FOR nombre-pros <ret>	$ LINK nombre-pros <ret>
PASCAL	$ PAS nombre-pros <ret>	$ LINK nombre-pros <ret>
COBOL	$ COB nombre-pros <ret>	$ LINK nombre-pros <ret>
COBOL/ANSI	$ COB/ANSI n-pros <ret>	$ LINK nombre-pros <ret>

Después de realizar el paso de programa escrito a programa objeto pueden aparecer errores, en este momento se deben corregir para poder correr el programa.

CURSOS: "INTRODUCCION AL SISTEMA VAX - 11/780" Y
"OPERACION Y ADMINISTRACION DEL SISTEMA VAX/VMS"
DEL 29 DE ABRIL AL 13 DE MAYO
MEXICO, D.F.



CORRECCION Y EJECUCION DE LOS PROGRAMAS.

PROFESORES:

ING. EDUARDO S. JALLATH CORIA.
ING. ALEJANDRO JIMINEZ GARCIA.
ING. SOCRATES A. MUÑIZ ZAFRA.
ING. HUMBERTO SANCHEZ SANDOVAL.

MAYO DE 1985.

4
CORRECCION Y EJECUCION DE LOS PROGRAMAS

4.1. EDITOR DE ERRORES:

Al efectuarse el proceso de compilación, es probable que salgan una serie de mensajes de error, esto durante la fase de conversión de programa ESCRITO a programa OBJETO. Para realizar la corrección de dichos errores, se debe utilizar la instrucción EDT para pasar de COMANDOS DEL SISTEMA a COMANDOS DE EDITOR y llegar a estar dentro del EDITOR. Todo esto viene en la parte dos de este manual. La forma en que se utilizará el comando EDT es dando el nombre de programa y la extensión del programa que tenga los errores, todo esto bajo el siguiente formato:

```
$ EDT nombre-del-programa.extension <return>
```

Ya una vez que el usuario se haya posicionado dentro del EDITOR por lo descrito en la parte tres de este manual, la correción del programa se puede efectuar mediante los comandos del Editor en modo línea. El sistema VAX 11-780 tiene una serie de funciones las cuales le ayudan al usuario a realizar una versatil corrección de los programas, llamadas FUNCIONES EDITOR en modo caracter. Es conveniente que el usuario primero se familiarice con los comandos de modo línea antes de querer utilizar el modo caracter. Las mas elementales son la tecla DELETE, la cual indica borrado de derecha a izquierda, y las cuatro flechas con las cuales se puede mover sobre lo escrito en las cuatro direcciones. Es importante recordar el no dejar pegadas las flechas ya que son repetitivas y la velocidad de transmisión del sistema es muy rápida; esto puede originar que se destruyan los archivos. Las FUNCIONES EDITOR se describirán en el siguiente cuadro y se refieren al KEYPAD(teclado auxiliar situado a la derecha del teclado:

FUNCION	CARACTERISTICAS
PF 1	Permite efectuar la operación alterna de cada tecla.
PF 2	Despliega todas las funciones del KEYPAD (HELP)

PF 3	Se posiciona en el siguiente STRING que se haya pedido que busque o dá el mensaje cuando no se encontro.
PF 1 PF 3	Aparece un mensaje que dice Search for,, se puede dar cualquier STRING que se desea buscar y necesita darle la dirección de búsqueda.
PF 4	Borra línea por línea, estando situado al principio de ésta
PF 1 PF 4	Imprime la última línea que se haya borrado con PF 4 estando situado al lado izquierdo del display
,	Borra caracter por caracter de izquierda a derecha
PF1 ,	Imprime en el display el último caracter que se haya borrado con ,
_	Borra palabra por palabra de izquierda a derecha, esto es de blanco a blanco, los caracteres entre espacios
	Imprime en el display la última palabra que [illegible] con _
/	Avanza a la siguiente página, y depende de [illegible]ción actual encendida
PF 1 7	Permite ejecutar un comando del editor en modo línea, hay que oprimir <enter>
8	Avanza secciones de 16 renglones
PF 1 8	El texto seleccionado es formateado para llenar el número de columnas que se fije con SET WRAP (modo línea).
9	El párrafo seleccionado es guardado al final del BUFFER
PF 1 9	Substituye el texto seleccionado por el texto contenido en el BUFFER.
4	Indica que todos los movimiento deseados son hacia el final del archivo

PF 1 4	Se sitúa en la parte inferior del archivo
5	Indica que todos los movimientos deseados son hacia el inicio del archivo
PF 1 5	Se sitúa en la parte superior del archivo
6	Corta el parrafo que se haya seleccionado y lo guarda en un BUFFER.
PF1 6	Imprime el último parrafo que se haya seleccionado y guardado en el BUFFER
1	avanza una palabra (en la dirección seleccionada
PF 1 1	Abre una línea donde esté el cursor para insertar texto.
2	Se posiciona al final de la línea donde se encuentre el cursor
PF 1 2	Borra todos los caracteres desde donde está el cursor hasta el fin de la línea
3	Avanza un caracter (en la dirección seleccionada)
PF1 3	Permite insertar un caracter ASCII
.	Inicia la selección de un párrafo que se quiera borrar o mover de lugar, abarcandolo con las flechas guia.
PF1 .	Cancela la selección del párrafo
ENTER	Permite que el comando de Editor modo línea indicado después de PF 1 7 sea ejecutado por la computadora
PF1 ENTER	substituye el STRING buscado por el texto en el BUFFER y busca la siguiente ocurrencia del STRING

Este diagrama en forma gráfica del KEYPAD es:

PF1	PF2	PF3	PF4
GOLD	HELP	FIND NEXT	DEL LINE
		find	und line
7	8	9	-
PAGE	SECTION	APPEND	DEL WORD
command	fill	replace	und word
4	5	6	,
ADVANCE	BACKUP	CUT	DEL CHAR
bottom	top	paste	und char
1	2	3	
WORD	EOL	CHAR	ENTER
chng case	del eol	specins	
0		.	
LINE		SELECT	
open line		reset	subs

Al estar en Editor modo caracter pueden aparecer en el texto los mensajes del sistema, para borrarlos hay que teclear CTRL/W.

4.2. VERSIONES .LIS/LIS:

Para ver los resultados de la compilación del programa, el ususario tiene una gran ventaja, las versiones .LIS. Estas versiones LIS se crean agregando /LIS en el proceso de compilacion, lo cual crea en el directorio los archivos NOMBRE-PROG.LIS las cuales se pueden ver mediante la instrucción TYPE.

Este cuadro es:

PROGRAMA ESCRITO	PROGRAMA OBJETO
BASIC	$ BAS/LIS nombre-pros
FORTRAN	$ FOR/LIS nombre-pros
PASCAL	$ PAS/LIS nombre-pros
COBOL	$ COB/LIS nombre-pros
COBOL/ANSI	$ COB/ANSI/LIS n-pros

COMANDO TYPE. Lo utiliza el ususario cuando está dentro de DCL.

Formato General:

$ TYPE nombre-del-programa.extensión;versión<return>

Esta instrucción despliega los archivos (programas, datos, resultados) en la pantalla sin ningún efecto para ellos; NO ES EL COMANDO TYPE DEL EDITOR.

Si el ususario pide un TYPE de un archivo .LIS generado en la compilación de su programa, aparecerá el listado del programa y los errores de sintaxis, si los hubo, intercalados en las líneas del listado. Cuando sean demasiados errores conviene imprimir el listado.

4.3. DANDO DATOS POR TERMINAL.

Cuando el usuario manda ejecutar su programa (imagen o módulo .EXE), deberá proporcionar los datos mediante la terminal y debe hacerlo siguiendo el formato especificado en el programa.

La forma de correr el programa es:

$ RUN nombre-del-programa <return>

Si el programa es en BASIC aparecerá una interrogación(?) pidiendo los datos, si es FORTRAN no aparecerá nada antes del cursor.

Si se desea interrumpir la ejecución del programa se puede hacer mediante las teclas CTRL-Y, CTRL-Z o CTRL-C dependiendo de la situación que se esté manifestando.

Para saber el tiempo de procesador que está consumiendo el programa el usuario deberá teclear CTRL-T.

4.4. CREACION DE ARCHIVOS DE DATOS

Cuando el programa requiere muchos datos es mejor crear archivos de datos (.DAT) mediante el editor para, posteriormente, asignarlos al canal de entrada de datos de tu programa. El formato para crearlo es:

$ EDT nombre-de-archivo.DAT <return>

Y crear el archivo que contenga los datos siguiendo los formatos indicados en el programa. Posteriormente se deberá asignar a SYS$INPUT mediante la instrucción ASSIGN que se verá posteriormente.

CURSOS: "INTRODUCCION AL SISTEMA VAX - 11/780" y
"OPERACION Y ADMINISTRACION DEL SISTEMA VAX/VMS"
DEL 29 DE ABRIL AL 13 DE MAYO
MEXICO, D.F.

OTRAS INSTRUCCIONES Y MODOS.

PROFESORES:

ING. EDUARDO S. JALLATH CORIA.
ING. ALEJANDRO JIMENEZ GARCIA.
ING. SOCRATES A. MUÑIZ ZAFRA.
ING. HUMBERTO SANCHEZ SANDOVAL.

MAYO DE 1985.

OTRAS INSTRUCIONES Y MODOS

5.1. OTRAS INSTRUCIONES:

Existe una gama muy amplia de comandos de sistema que se pueden usar en el sistema VAX 11-780 tales como SHOW, SET, etc..
Todos estos están resumidos en una biblioteca a la que se tiene acceso mediante el comado HELP, en el cual están los formatos de los comandos de DCL.
Algunas instrucciones están restringidas a la mayoría de los usuarios pero aun así hay gran versatilidad.

El formato de HELP es:

$ HELP <return>

A continuación aparece un desplegado con todos los comandos que existen en DCL y se puede pedir su sintaxis escribiendo el comando después del mensaje que dice Topic?.

La computadora dará al usuario una pequeña explicación del comando y proporcionará información adicional que puede explicar. Si así se desea, el usuario deberá escribirlo despues de la palabra Subtopic?

La forma de salirse del HELP es mediante <return> sucesivos hasta que aparezca el signo de $.

Es conveniente que las primeras veces que se use la VAX se compenetre en el uso de este comando de DCL.

$ HELP <return>

Information available:

ACCOUNTING	ALLOCATE	ANALYZE	APPEND
ASSIGN	ATTACH	BACKUP	BASIC
BLISS	CANCEL	CC	CLOSE
COBOL	CONTINUE	CONVERT	COPY
CORAL	CREATE	DBO	DDL
DEALLOCATE	DEASSIGN	DEBUG	DECK
DEFINE	DELETE	DEPOSIT	DIFFERENCES
DIRECTORY	DISMOUNT	DUMP	EDIT
EOD	EOJ	Errors	

```
   EXAMINE        EXIT           FDL            FORTRAN
   GOTO           HELP           IF             INITIALIZE
   INQUIRE        JOB            Lexical
   LIBRARY        LINK           Login          LOGOUT
   MACRO          MAIL           MCR            MERGE
   MESSAGE        MONITOR        MOUNT          ON
   OPEN           PASCAL         PASSWORD       PATCH
   PHONE          PLI            PRINT          Procedure
   PURGE          Queues         READ           RENAME
   REPLY          REQUEST        RMS            RTL
   RUN            RUNOFF         SEARCH         SET
   SHOW           SORT           SPAWN
   Specify        START          STOP           SUBMIT
   Symbol_Assign                 SYNCHRONIZE
   System         TECO           TYPE           UNLOCK
   WAIT           WRITE

Topic?  dir  <return>

DIRECTORY

   Provides a list of files or information about a
   file or group of files.

   Format:

        DIRECTORY   [file-spec[,...]]

  Additional information available:

  Parameters Qualifiers
   /BEFORE      /BRIEF       /COLUMN      /CREATED
   /DATE        /EXCLUDE     /EXPIRED     /FULL
   /HEADING     /MODIFIED    /OUTPUT      /OWNER
   /PRINTER     /PROTECTION               /SINCE
   /SIZE        /TOTAL       /TRAILING    /VERSIONS

DIRECTORY Subtopic? /full  <return>

DIRECTORY

  /FULL

   Lists full file attributes with each file.

   The /FULL qualifier overrides the default brief listing
format.


DIRECTORY Subtopic?  <return>
```

Topic? <return>

$

5.2. BASIC INTERACTIVO

Una de las dos formas de trabajar en lenguaje BASIC en la VAX es mediante BASIC interactivo. Para lograr utilizar este BASIC es necesario invocarlo mediante el comando de DCL

$ BASIC <return>.

Este modo de trabajar en BASIC puede ser similar al utilizado en las microcomputadoras, con la diferencia de que éste BASIC compila el programa aunque no guarde el código objeto en el directorio. Esto es muy útil cuando se está probando o depurando un programa.

Formato:

$ BASIC <return>

A partir de este momento se está en el Ambiente BASIC y el sistema enviará un mensaje que dice:

VAX-11 BASIC V2.0

Ready

Después de esto, el usuario podrá dar un conjunto de instrucciones entre las que estan :

HELP de Basic
OLD
NEW

OLD <return>

Permite traer al ambiente BASIC un archivo que existe en el directorio y que debe ser del tipo .BAS. Este archivo contiene un programa en lenguaje BASIC que se quiera correr.

NEW <return> Hace que BASIC borre el archivo existente en el ambiente y que pida el nombre del nuevo programa BASIC que se va a crear. Este nombre debe tener entre uno y nueve caracteres

Formato:
NEW <return>

New file name-----Nombre_del_programa

Se pone el Nombre_del_programa y no hace falta ponerle la extension.

Cuando el programa es OLD la computadora envia el mensaje de Ready y cuando es NEW deja el cursor en la parte inferior del Display esperando que el usuario empiece a escribir el programa.

Cuando se termine de escribirlo podrá correrlo inmediatamente después mediante la instrucción RUN o RUNNH. Si existe algún tipo de error el BASIC lo indicará sin haber iniciado la ejecución.
Este error se podrá corregir con el comando EDIT del ambiente BASIC y tratar de ejecutar de nuevo el programa.

La instruccion LIST despliega el programa en la terminal.

Para graber el programa en disco es necesario que se le de la instrución SAVE para que se almacene en el directorio.

Para salir del ambiente BASIC se deberá dar la instrucción EXIT.

Ejemplo:

$ BASIC <return>

VAX-11 BASIC V2.0

Ready<LF>

help

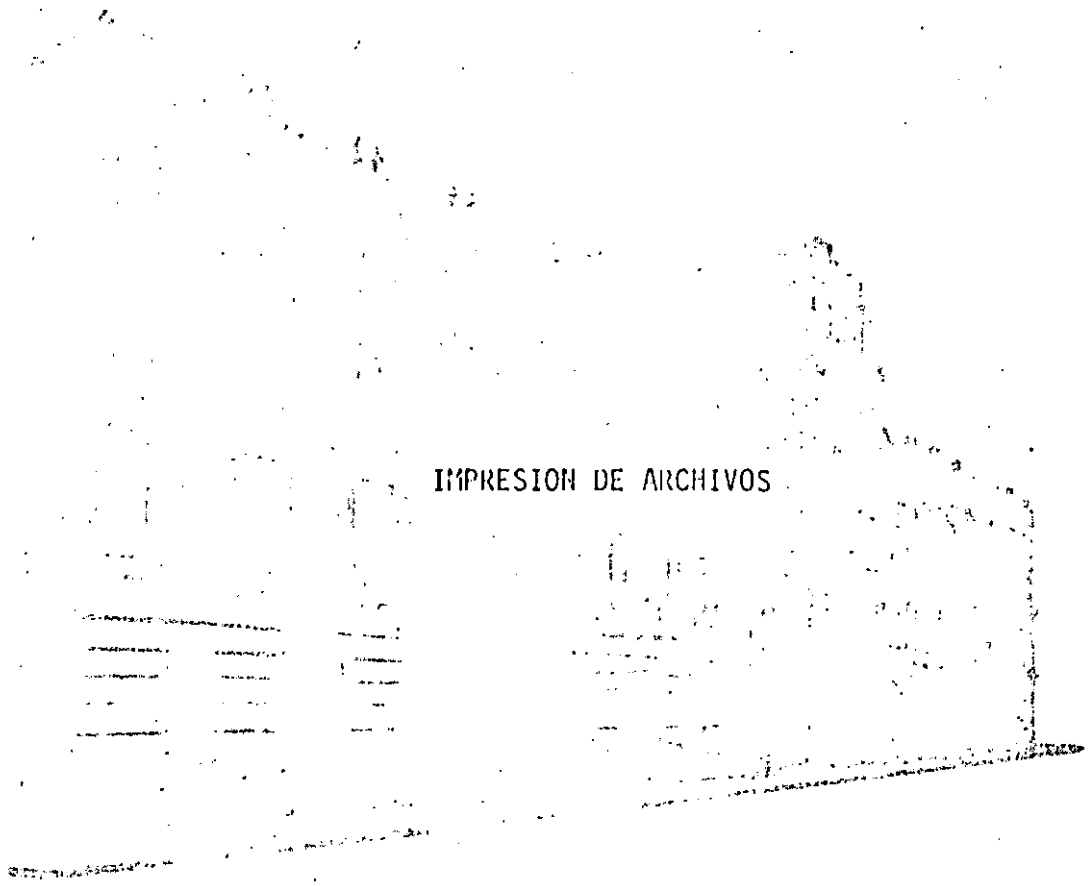
HELP

The HELP command displays on-line information about BASIC statements, commands, directives, functions, conventions, and other topics. Type HELP to see a list of topics. Then enter a subtopic for more information. If you type a question mark in response to the prompt for a topic, BASIC displays the list of available topics.

```
  Additional information available:

  ARRAYS       CHARACTER   COMMANDS
  COMMENTS     CONSTANTS   CONVENTIONS
  DATA_TYPES   DIRECTIVES  ERRORS
   EXPRESSIONS              FUNCTIONS  HELP
  IMMEDIATE    LABELS      LINE
  MODIFIERS    QUALIFIERS  STATEMENTS VARIABLES

Topic? stat

STATEMENTS

    Statements assign values, perform I/O, transfer
program control, and so forth. Program statements
are associated with a line number and stored for later
execution. A statement starting in the first column and
having no line number is executed as an immediate
mode statement. For additional information on
immediate mode statements, type 'HELP IMMEDIATE'.

    Statement modifiers are keywords that qualify or
restrict a statement. For help on modifiers, type 'HELP
MODIFIERS'.


  Additional information available:

  CALL          CHAIN         CHANGE      CLOSE
  COMMON        DATA          DECLARE
  DEF           DELETE        DIMENSION   END
  EXIT          EXTERNAL      FIND
  FNEND         FNEXIT        FOR         FREE
  FUNCTION      FUNCTIONEND
  FUNCTIONEXIT                GET         GOSUB
  GOTO          IF            INPUT
  ITERATE       KILL          LET         LINPUT
  LSET          MAP           MARGIN
  MAT           MOVE          NAME_AS     NEXT
  NOMARGIN      ON            OPEN
  OPTION        PRINT         PUT         RANDOMIZE
  READ          RECORD        REM
  REMAP         RESTORE       RESUME      RETURN
  RSET          SCRATCH       SELECT
  SLEEP         STOP          SUB         SUBEND
  SUBEXIT       UNLESS        UNLOCK
  UNTIL         UPDATE        WAIT        WHILE

STATEMENTS Subtopic? if

  IF
```

```
     The IF statement evaluates a conditional expression
and
     transfers  program  control, depending on  the
resulting
     value.

     Format

     Conditional:

               {   statement...}}
               {THEN {lin-num  }}[      {lin-num      }
IF cond-exp    {                 }[ELSE {statement..}] [END IF]
               {GOTO target      }

     Statement Modifier:

          statement IF cond-exp

     Examples

     100       IF A > 0
               THEN PRINT A
               ELSE PRINT "A IS NOT GREATER THAN 0"
               END IF

     200       PRINT A IF A > 0

STATEMENTS Subtopic?  <return>

Topic?  <return>


Ready <LF>

exit

$
```

CURSOS: "INTRODUCCION AL SISTEMA VAX - 11/780" Y
"OPERACION Y ADMINISTRACION DEL SISTEMA VAX/VMS"
DEL 29 DE ABRIL AL 13 DE MAYO
MEXICO D.F.



IMPRESION DE ARCHIVOS

PROFESORES:

ING. EDUARDO S. JALLATH CORIA.
ING. ALEJANDRO JIMENEZ GARCIA.
ING. SOCRATES A. MUÑIZ ZAFRA.
ING. HUMBERTO SANCHEZ SANDOVAL.

MAYO DE 1985.

IMPRESION DE ARCHIVOS

6.1. INSTRUCCION PRINT Y COPY

LA instrucción PRINT permite asignar a la impresora archivos y programa para su impresión. Este es una instrucción de COMANDOS DEL SISTEMA(DCL) por lo tanto se debe dar cuando este el signo de $.

El formato general es:

$ PRINT nombre_del_programa_o_archivo.extension;versión

Después el sistema genera un aviso de entrada de impresión:

Job nnn entered on SYS$PRINT

Donde nnn es el numero de listado de impresión que le asignó al programa.

Cuando el listado ha terminado de salir por al impresora, la computadora enviará el mensaje correspondiente.

Si se desea escribir varios archivos bajo la misma carátula de salida lo que se tiene que hacer es asignarlos despues de la instrucción PRINT separados por comas(,) o por mas(+).

El formato es:

$PRINT archivo_uno,archivo_dos+archivo_tres,...archivo_n <return>

Si no se escribe la versión, la computadora entiende que se quiere mandar a imprimir la última versión.

Está restringido por sistema y no tiene sentido mandar a imprimir archivos del tipo .OBJ y .EXE.

INSTRUCCION COPY

Esta instrucción pertenece a las instrucciones de DCL mediante la cual se puede asignar a un solo archivo un conjunto de ellos , o copiar archivos de un subdirectorio a otro.

El formato es:

```
$COPY <return>
$_from:  que_archivo_vas_a_asignar <return>
$_to:  a_que_archivo_lo_vas_a_asignar <return>
```

En algunos casos el sistema enviará un mensaje de incompatibilidad de archivos pero no tiene trascendencia.

En la parte de $to: del formato se debe asignar un nombre con la extensión que se desee, este nuevo archivo estará en el directorio personal mediante el nombre que se designó y contendrá todos los archivos que se determinaron en el $from: .

6.2. ASIGNACION DE INPUT Y OUTPUT:

Para poder entender perfectamente esta parte del manual es necesario hacer algunas aclaraciones:

Normalmente la salida de resultados de la corrida de un programa y la entrada de datos para poder efectuar la corrida de éste, está asignada a pantalla. Lo que se va a realizar en este caso es cambiar esa asignacion normal del sistema, de pantalla a una asignación de un archivo personal en disco.

El sistema de salida es el SYS$OUTPUT y el de entrada es el SYS$INPUT.

La aplicación práctica es el almacenar los resultados de un programa en un archivo SYS$OUTPUT y el tener los datos de un programa grabados en un archivo mediante la instrucción SYS$INPUT.

Las instrucciones de COMANDOS DE SISTEMA: ASSIGN SYS$OUTPUT y ASSIGN SYS$INPUT nos permiten crear archivos con los resultados de nuestros programas y asignar los datos contenidos en un archivo .DAT a un programa.

Toda instrucción ASSIGN debe llevar un respectivo DEASSIGN, esto es: debemos asignar de entrada SYS$INPUT o de salida SYS$OUTPUT y desasignar posteriormente con un DEASSIGN.

El formato general de SYS$OUTPUT:

```
$ASSIGN nombre_prog.RES SYS$OUTPUT <return>
$RUN nombre_del_programa <return>
$DEASS SYS$OUTPUT <return>
```

Si se desea se puede poner .LIS en lugar de .RES, es indiferente.

El archivo a imprimir es nombreprog.RES (o nombre.LIS si así se indicó) y aparecerá en el directorio y se puede mandar imprimir. en el se encuentran los resultados..

Formato del SYS$INPUT:

```
$ASS nombre_de_archivo.DAT SYS$INPUT <return>
$RUN nombre_de_programa
$DEASS SYS$INPUT
```

El programa correrá con los datos contenidos en nombre_de_archivo.DAT.

Hay que aclarar que cuando se este en modo SYS$OUTPUT no aparece ningun resultado en pantalla sino que se enviarán a un archivo de salida.

CURSOS: "INTRODUCCION AL SISTEMA VAX - 11/780" Y
"OPERACION Y ADMINISTRACION DEL SISTEMA VAX /VMS"
DEL 29 DE ABRIL AL 13 DE MAYO.
MEXICO, D.F.

MANTENIMIENTO DE LA BIBLIOTECA EN ORDEN

PROFESORES:

ING. EDUARDO S. JALLATH CORIA.
ING. ALEJANDRO JIMENEZ GARICA.
ING. SOCRATES A. MUÑIZ ZAFRA.
ING. HUMBERTO SANCHEZ SANDOVAL.

MEXICO, D.F.

MANTENIMIENTO DE LA BIBLIOTECA EN ORDEN

7.1. INSTRUCCIONES PURGE Y DELETE.

Para poder mantener la biblioteca en orden existen dos instrucciones de COMANDOS DEL SISTEMA muy importantdes.

INSTRUCCION PURGE:

Esta instrucción depura el directorio dejando unicamente la última versión de cada programa que se tenga en el directorio.

Hay que tener mucho cuidado al utilizar esta instrucción ya que muchas veces la versión que sirve no es la última y si se ejecuta la instruccion PURGE el sistema eliminará la versión útil.

El formato es:

$ PURGE <return>

Aparecerá el signo de $.

INSTRUCCION DELETE:

Esta instrucción permite borrar un determinado archivo o familia de ellos.

El formato es:

$ DELETE nombre_del_archivo.extension;versión <return>

Puede aparecer el signo de $ o un mensaje que indique el hecho de tratar de borrar un archivo que no existe. O que se tiene privilegios para borrar el archivo (como son los subdirectorios)

El asterisco (*) tiene la particularidad de generalizar como sigue:

$ DEL *.NNN;*

Esto borra todas las versiones con esa extension, sin importar el nombre y versión.

$ DEL *.*;N

Borra el archivos cuya versión sea la indicada

$ DEL NNNN.*;*

Borra todas las versiones y extensiones del archivo indicado.

Las instrucciones para borrar pueden ser las mas fáciles de ejecutar, pero tambien son las mas peligrosas ya que se puede llegar a borrar archivos o programas que posteriormente harán falta.

El uso del (*) como generalizador (wild card) es válido en cualquier instrucción de DCL relacionada con archivos. Otro caracter es (%) que generaliza un solo caracter.

Una instrucción que nos permite cambiar el nombre a archivos ya existentes es el $ RENAME nombre-viejo nombre-viejo.

CURSOS: "INTRODUCCION AL SISTEMA VAX - 11/780" Y
"OPERACION Y ADMINISTRACION DEL SISTEMA VAX /VMS"
DEL 29 DE ABRIL AL 13 DE MAYO.
MEXICO, D.F.

SALIENDO DEL SISTEMA

PROFESORES:

ING. EDUARDO S. JALLATH CORIA.
ING. ALEJANDRO JIMENEZ GARICA.
ING. SOCRATES A. MUÑIZ ZAFRA.
ING. HUMBERTO SANCHEZ SANDOVAL.

MAYO DE 1985.

8
SALIENDO DEL SISTEMA

8.1. VERSIONES .JOU E INSTRUCCION RECOVER:

Durante una sesión de trabajo en el sistema VAX 11-780 puede llegar a suceder una interrupción de corriente eléctrica, o que ocurra otro tipo de falla. Esto no tiene gran trascendencia si se está en DCL pero si se encuentra dentro del Editor y no se puede volver normalmente al Editor o a DCL entonces toda la sesión de Editor podría haberse perdido, pero la computadora la guarda en un archivo de tipo .JOU.

El archivo o programa afectado por la interrupción quedará en el directorio con una extensión .JOU lo cual viene de la palabra JOURNAL, este tipo de archivos trae una especie de resumen de las operaciones efectuadas sobre dicho archivo en la sesión de Editor. Las versiones .JOU no se deben accesar de la misma forma que los archivos normales que se describen en el comando $ EDT. Lo que se tiene que hacer es utilizar el calificador RECOVER para recuperar la sesion interrumpida.

El formato general es:

$ EDT/RECOVER nombre_del_archivo_orisinal <return>

Esto hará que la computadora reescenifique la sesión de Editor hasta el momento en que ocurrió la falla, esto es, la computadora hará de nuevo todas las operaciones que se realizaron en la sesión de Editor y lo dejará para que se continue.

8.2. INSTRUCCION LOGOFF:

Esta instrucción sirve para despedirse del sistema VAX 11-780, y se debe utilizar al finalizar la sesión de terminal.

Se puede utilizar el Comando LOGOFF, su abreviatura LOG o el símbolo LO.

Formatos:
$ LOGOFF <return>
$ LOG <return>
$ LO <return>
Aparecerá un mensaje de despedida del sistema y la conexion con VAX acaba de concluir.

CURSOS: "INTRODUCCION AL SISTEMA VAX - 11/780" Y
"OPERACION Y ADMINISTRACION DEL SISTEMA VAX /VMS"
DEL 29 DE ABRIL AL 13 DE MAYO.
MEXICO. D.F.

PROCEDIMIENTOS DE COMANDOS.

PROFESORES:

ING. EDUARDO S. JALLATH CORIA.
ING. ALEJANDRO JIMENEZ GARCIA.
ING. SOCRATES A. MUÑIZ ZAFRA.
ING. HUMBERTO SANCHEZ SANDOVAL.

MAYO DE 1985.

9

PROCEDIMIENTOS DE COMANDOS

9.1 SIMBOLOS

Los símbolos son nombres de variables, de 1 a 9 caracteres, en las cuales podemos guardar números reales o enteros o strings (que pueden ser comandos de DCL).

Los símbolos pueden ser locales a un procedimiento o pueden ser globales a todos los procedimientos y comandos ejecutados en la sesión. Para almacenar datos en los símbolos se deben seguir las siguientes reglas:

Para valores numéricos locales:

SIMBOLO = valor

Para valores numéricos globales:

SIMBOLO == valor

Para strings locales:

SIMBOLO := string

Para string globales:

SIMBOLO :== string

Mediante estos símbolos se pueden redefinir las instrucciones para simplificar los procesos, pero sólo son válidos durante la sesión y únicamente en la clave del usuario. Ejemplo:

AUXILIO:==HELP

Cada vez que se escriba AUXILIO, la computadora ejecutará el comando HELP.

Si dentro del nombre del símbolo se inserta un asterisco, entonces todos los caracteres que le siguen son opcionales

Ejemplo
AUX*ILIO:==HELP
Con solo teclear AUX se ejecutará el comando HELP.

A nivel DCL se permite ejecutar operaciones con los símbolos. Estas operaciones son: suma, resta, producto y división para los valores numéricos y búsqueda, extraccion y concatenación, y substitución de contenido en los strings.

9.2) PROCEDIMIENTOS DE COMANDOS

Un procedimiento de comandos es un archivo que contiene una secuencia de instrucciones de DCL. Este archivo se crea con el Editor y debe ser del tipo .COM. Cada instrucción debe estar precedida por el símbolo de $. Para continuar una instrucción en otra línea hay que escribir un guión al final de la línea y continuar en la siguiente sin poner el signo de $.

Para documentar el procedimiento de comandos se utilizará un signo de admiración (!) para indicar que el resto de la línea es un comentario.

Se deben utilizar los comandos completos, esto es, sin abreviaturas, y se recomienda el uso de la sangría en las líneas para mejorar la legibilidad y comprensión del procedimiento.

Para ejecutar un procedimiento de comandos es necesario preceder el nombre del archivo de comandos, del signo arroba (@) por ejemplo: @LOGIN.

Se puede ejecutarlo también mediante un SUBMIT, que ocasiona que el proceso entre a la cola de BATCH. También se puede ejecutarlo desde otro procedimiento de comandos.

Una manera sencilla de hacerlo, sobre todo si el nombre es muy largo (incluyendo dispositivo, directorio, subdirectorio, etc.) es utilizando un símbolo como sinónimo del procedimiento. Este símbolo puede definirse en el LOGIN.COM.

Para verificar la ejecución del procedimiento se pueden utilizar los comandos SET VERIFY al inicio y SET NOVERIFY al final para que todos los comandos sean desplegados en la terminal

Control de Entrada/Salida de los procedimientos de Comandos. Cuando se inicia una sesión, el sistema operativo (computadora) ejecuta un proceso en el cual se establecen las equivalencias iniciales de los siguientes nombres lógicos:

SYS$INPUT	Es el canal de entrada de datos.
SYS$OUTPUT	Es el canal para desplegar información.
SYS$ERROR	Es el canal para desplegar los mensajes de Error.
SYS$COMMAND	Es el canal para introducir comandos.
SYS$DISK	Es el canal donde se encuentran alojados los archivos.
SYS$LOGIN	Contiene el dispositivo y directorio iniciales al entrar en sesión.

Cuando se inicia una sesión, SYS$INPUT se encuentra asignado a la terminal en la cual se está trabajando. Cuando se ejecuta un procedimiento de comandos, se establece una nueva equivalencia para el nombre lógico SYS$INPUT, la cual será el propio archivo que contiene el procedimiento de comandos. Si se anidan procedimientos de comandos, es decir si es ejecutado un procedimiento dentro de otro procedimiento de comandos, SYS$INPUT es asignado al archivo que corresponde al procedimiento que se esté ejecutando en ese momento.

SYS$COMMAND es hecho equivalente al nivel de comandos en que se encuentre: si se ejecuta un procedimiento de comandos interactivamente, este nombre lógico es asignado a la terminal, mientras que si ejecutado desde batch es asignado a la lectora de tarjetas.

Otra forma de mandar ejecutar procedimientos de comandos es mediante el SUBMIT seguido del nombre del archivo. Esta ejecución entrará a la cola de BATCH.

UTILIZANDO SIMBOLOS EN PROCEDIMIENTOS DE COMANDOS.

Podemos definir un símbolo que contenga el nombre de un archivo y utilizarlo en los comandos encerrado entre apóstrofes (') que le indica a la computadora que efectué la substitución del símbolo por su valor.

El contenido de los símbolos puede tener hasta 255 caracteres y generalmente son nombres de archivos.

En un símbolo se pueden almacenar expresiones de tipo string o numéricas. Por ejemplo:

```
$ CODE = 4 + F$STRING('6') - A
```

donde A es un símbolo previamente definido.

9.3) OPERACIONES Y RELACIONES

Las expresiones pueden ser:

expresión	modo
valor entero	entero
valor string	string
función entera	entero
función string	string
símbolo entero	entero
símbolo string	string
+, - o .NOT. valor	entero
valor .AND. o .OR. valor	entero
string + o - string	string
entero + o - cualquier valor	entero
valor + o - entero	entero
valor * o / cualquier valor	entero
valor (comparación string) valor	entero
valor (comparación aritmética) valor	entero

En la tabla anterior cualquier 'valor entero' o 'valor' es un valor de caracteres string.

Existen algunas funciones para formar las expresiones y entre las más usadas están:

F$EXTRACT	extrae un substring
F$INTEGER	convierte a número un string
F$LENGTH	nos indica la longitud de un string
F$LOCATE	localiza un caracter en un string
F$LOGICAL	traduce un nombre lógico (definido previamente)
F$STRING	convierte un resultado entero a string.

Para asignar expresiones string a símbolos se usa el formato símbolo = expresión string.

Para asignar expresiones enteras a símbolos se emplea el siguiente formato: símbolo = expresión entera.

Los operadores en las expresiones son:

tipo operador	precedencia	operación
lógico - .OR.	1	o
.AND.	2	y
.NOT.	3	complemento
comparac. .EQ.	4	= aritmético
aritmét. .GE.	4	>= "
.GT.	4	> "
.LE.	4	<= "
.LT.	4	< "
.NE.	4	<> "
comparac. .EQS.	4	= strins
strins .GES.	4	>= "
.GTS.	4	> "
.LES.	4	<= "
.LTS.	4	< "
.NES.	4	<> "
Operad. +	5	suma aritmética
aritm. -	5	diferencia "
+	7	signo positivo
-	7	signo negativo
*	6	producto aritmético
/	6	cociente "
Operad. +	5	concatenación strins
strins -	5	reducción strins

Nota: la prioridad es 1 = más baja y 7 = la más alta.

Para utilizar los operadores lógicos se pueden usar valores enteros o sus correspondientes binarios si se precede de un % y una %. Ejemplo:

```
A = %X1000 .OR. %X0001
A =   8    .OR.    1
```

en ambos casos el resultado es 9.

Para controlar la ejecución de los comandos existen una serie de instrucciones que son:

```
IF expresión de comparación THEN comando
GO TO etiqueta
EXIT
STOP
```

Las etiquetas son nombres seguidos por dos puntos (:)

Para el manejo de los errores se utiliza el comando ON nivel de severidad THEN comando

Los errores pueden ser WARNING, ERROR, SEVERE_ERROR tal como se definen en el manual de DCL.

Un error simulado se obtine con CONTROL_Y de la forma siguiente ON CONTROL_Y THEN GOTO FIN donde FIN es una etiqueta. Cuando se oprimen en forma simultánea CTRL y la Y entonces el procedimiento de comandos se dirige a la etiqueta FIN.

Para permitir el uso de CTRL/Y se debe escribir al inicio $SET CONTROL=Y y para desabilitarlo se escribe $SET NOCONTROL=Y pero debe tenerse mucho cuidado en su uso porque si el porceso se queda en un 'loop' no hay forma de detenerlo

Para escribir información se utiliza el $WRITE. El WRITE debe ir acompañado del nombre del dispositivo o archivo en el cual queremos imprimir la información. Si se desea desplegar algo en la pantalla se debe escribir $ WRITE SYS$OUTPUT expresión

Para leer información vía terminal se debe utilizar el comando $INQUIRE nombre del símbolo [promt]. El promt es el mensaje que la computadora imprimirá al realizar el INQUIRE.

Un ejemplo de esto puede ser:

$INQUIRE DATO Nombre de tu Programa.

Note que el promt va sin comillas.

Para profundizar más en esto, sería conveniente que tomaran un curso de DCL.

9.4) LOGIN

El LOGIN.COM es un archivo de comnados que el sistema ejecuta al entrar en sesión. En el se recomienda definir símbolos con los comandos más usados en nuestro diario trabajo, en la VAX.

Un ejemplo del uso de procedientos de comandos es:

```
$ SET NOVERIFY
$ ! COMPILA  LIGA  CORRE
$ WRITE SYS$OUTPUT "                    COMPILA  LIGA  COOORREEE
!!!"
$ WRITE SYS$OUTPUT ""
$ PARAM1:
$ IF P1 .NES. "" THEN GOTO ELSE1
$     INQUIRE P1 PROGRAMA
$     GOTO PARAM1
$ ELSE1:
$     PROGRAMA:=='P1'
$ ENDIF1:
$ PARAM2:
$ IF P2 .NES. "" THEN GOTO ELSE2
$     INQUIRE P2 LENGUAJE
$     GOTO PARAM2
$ ELSE2:
$     LENGUAJE:=='P2'
$ ENDIF2:
$ CADENA:='LENGUAJE'
$ DEPURA:=""
$ LOOP:
$     LONGITUD='F$LENGTH(CADENA)'
$     DESPLA='F$LOCATE("/",CADENA)'
$     IF DESPLA .EQ. LONGITUD THEN GOTO ENDLOOP
$         DIFER=LONGITUD-DESPLA
$         AUXILIO:='F$EXTRACT(DESPLA,DIFER,CADENA)'
$         CHARACTER:='F$EXTRACT(1,2,AUXILIO)'
$         IF CHARACTER .EQS. "DE" THEN GOTO ELSE4
$             DIFER=DIFER-1
$             CADENA:='F$EXTRACT(1,DIFER,AUXILIO)'
$             GOTO ENDIF4
$         ELSE4:
$             DEPURA:="/DEBUG"
$             GOTO ENDLOOP
$         ENDIF4:
$         GOTO LOOP
$ ENDLOOP:
$ !
$ ! COMPILACION
$ !
$ WRITE SYS$OUTPUT "Compilando"
$ 'LENGUAJE' 'PROGRAMA'
$ !
$ ! LIGADO
$ !
$ LONGITUD='F$LENGTH(PROGRAMA)'
$ DESPLA='F$LOCATE(".",PROGRAMA)'
$ IF DESPLA .EQ. LONGITUD THEN GOTO LIGA
$       PROGRAMA:='F$EXTRACT(0,DESPLA,PROGRAMA)'
```

```
$ LIGA:
$ WRITE SYS$OUTPUT "Ligando"
$ LINK'DEPURA' 'PROGRAMA'
$ !
$ ! EJECUTA
$ !
$ ASSIGN/USERMODE SYS$COMMAND SYS$INPUT
$ WRITE SYS$OUTPUT "Corriendo !!!!!!!!!!!"
$ MAXFOR = 50
$ I = 1
$ FOR:
$ IF I .EQ. MAXFOR THEN GOTO ENDFOR
$       I = I+1
$       GOTO FOR
$ ENDFOR:
$ RUN 'PROGRAMA'
```

El siguiente programa define la posición de los tabuladores, limpia la pantalla y despliega el directorio.

```
$WRITE SYS$OUTPUT "<ESC>[2J"
$WRITE SYS$OUTPUT "<ESC>[;f"
$WRITE SYS$OUTPUT "<ESC>[3g"
$WRITE SYS$OUTPUT "<ESC>[1;7H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[7;10H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[10;13H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[13;17H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[17;21H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[21;25H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[25;29H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[29;33H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[33;37H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[37;41H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[41;45H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[45;49H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[49;53H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[53;57H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[?;4;6;8h"
$LISTADO:==SET TERM/WIDTH=132
$PROGRAMA:==SET TERM/WIDTH=80
$Z:==DIRECTORY
$SUB:==SET DEF [CLAVE.SUBDIR]
$BAS:==SET DEF [CLAVE.BASIC]
$Z80:==SET TERM/WIDTH=80
$DIR
```

El caracter <ESC> resulta de oprimir 2 veces la tecla ESC, y se debe estar en Editor modo caracter.

AYUDAS

En el sistema existen una serie de ayudas para hacer el trabajo más sencillo. Estas ayudas son:

10.1. MAIL

El MAIL nos permite dejarle mensajes a otros usuarios para que cuando entre en sesión la computadora le avise que tiene un recado y lo lea; si el usuario está en sesión cuando mandas el mensaje, la computadora le indicará en ese momento que tiene un mensaje.

Para utilizar esta ayuda lo primero que debe hacerse es teclear MAIL <return>, y utilizar los comandos propios del MAIL como son:>SEND, >READ, >DELETE, >HELP, >EXIT, etc.

10.2. PHONE

La ayuda del PHONE es para entablar un diálogo con otros usuarios que estén en sesión en ese momento. Para invocarlo es necesario teclear PHONE <return> y utilizar los comandos del PHONE cómo son %DIAL, %ANSWER, %HANGUP, %DIRECTORY, etc.

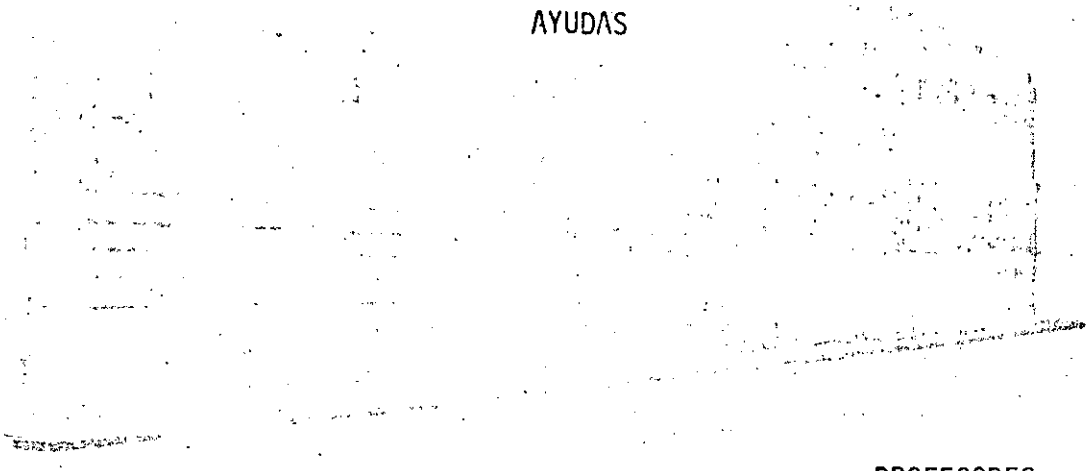
10.3. DEBUGGER

El Debugger o depurador permite ver el contenido de ciertas variables durante una corrida y, si es necesario, modificar valores. Para poder utilizarlo es necesario compilar y ligar el programa con los calificadores /DEBUG y al dar el comando RUN el programa no inicia la corrida sino que la computadora le cede el control al Debugger (promt= DBG>). Los comandos básicos son: GO, SET BREAK, EVALUATE, CANCEL BREAK, etc.

Se recomienda el estudio de estás facilidades en el instructivo correspondiente (sobre todo el Debugger) o acudir al HELP de cada uno de ellos.

CURSOS: "INTRODUCCION AL SISTEMA VAX - 11/780" Y
"OPERACION Y ADMINISTRACION DEL SISTEMA VAX /VMS"
DEL 29 DE ABRIL AL 13 DE MAYO.
MEXICO, D.F.

AYUDAS



PROFESORES:

ING. EDUARDO S. JALLATH CORIA.
ING. ALEJANDRO JIMENEZ GARICA.
ING. SOCRATES A. MUÑIZ ZAFRA.
ING. HUMBERTO SANCHEZ SANDOVAL.

MAYO DE 1985.

11
PROTECCIONES

Para proteger los archivos en una forma 'permanente' se puede valer de cintas magnéticas o de discos flexibles (diskettes). Estos tipos de almacenamientos van a permitir almacenar más información de la que se puede tener en nuestra área de disco.

11.1. CINTAS

Las cintas magnéticas son almacenamientos de tipo secuencial, las hay de 600, 800 y 1200 ft de longitud y se pueden grabar con una densidad de 800 ó 1600 bpi (bits por pulgada). El nombre que se le asigna al dispositivo inicia con las siglas MT, en la configuración actual, la cinta es la unidad'MTA0:.

11.2. DISKETTES

Los diskettes o Floppy Disk es un almacenamiento de acceso directo, ' permite tener el directorio y subdirectorios como si se estuviera en el disco. El nombre del dispositivo es DYA0: ó DYA1: dado que hay dos unidades. En el Floppy (que es de 8' pulgadas) se puede grabar a densidad sencilla o a densidad doble lo que proporciona 256 Kb y 512 Kb respectivamente.

No todos los discos de 8 pulgadas sirven, por lo que antes de adquirirlos deben preguntar las marcas y modelos de éstos que le sirven a la VAX.

11.3. COMANDOS

Para poder utilizar las cintas o los floppys es necesario inicializarlos, esto es darles una serie de atributos, que se la graben para que la computadora sepa, cada vez que los utilicemos, que características tienen. Estos atributos incluyen: un nombre, la densidad y la etiqueta.

Cada vez que se utilicen estos medios de almacenamiento es necesario 'adueñarse' de la unidad para que ningún otro usuario pueda trabajar con la cinta o diskette.

Después de realizada la operación, es necesario indicarlo a la máquina que 'monte' el medio de almacenamiento, ésto es, que permita el acceso porque aunque físicamente esté

colocado, lógicamente aún no está disponible hasta haberlo 'montado'.

Si el dispositivo es un Floppy es necesario posicionarse en el con el $SET DEF DYAx:, y luego crear los subdirectorios para posteriormente irse a ellos.

Cuando es cinta bastará ubicarnos en ella con el $SET DEF.

En este momento es cuando se puede realizar las operaciones de protección de disco al medio, mediante el $COPY, ó 'bajar' archivos del medio al disco, también con $COPY. Es necesario proporcionar los nombres completos (incluyendo dispositivos).

Después de esto hay que 'desmontar' y liberar el dispositivo.

Los Comandos de DCL necesarios son:

$INITIALIZE nombre-del-dispositivo[:] etiqueta-del-volumen

donde el calificador principal es /DENSITY=valor

donde valor debe ser 800 o 1600 para cintas o
SINGLE O DOUBLE para Floppy.

Esto es necesario hacerlo solo una vez, o cada vez que se desee borrar todo el contenido del medio de almacenamiento.

$MOUNT dispositivo [etiqueta] [nombre-lógico]

y sus calificadores principales son:
/FOREIGN
/DENSITY
/BLOCKSIZE
/RECORDSIZE
que se usan principalmente para cintas que vienen o van hacia otros equipos que no sean VAX.

$ALLOCATE dispositivo [nombre-lógico]

para lograr la privacía en el acceso.

$COPY archivo-origen archivo-destino

donde archivo-origen es el nombre completo (incluyendo el

dispositivo) del archivo que se desea copiar; es permitido utilizar los 'wild cards'. Y el archivo-destino es el nombre del archivo que se quiere que reciba el(los) archivo(s) que se copia(n). Cabe hacer notar que se puede omitir el dispositivo en el que nos encontramos.

Para saber que archivos se tienen, se puede pedir el Directorio con el comando $DIRECTORY.

$DISMOUNT dispositivo

Ocasiona que el medio de almacenamiento masivo sea liberado o desmontado.

$DEALLOCATE dispositivo

Sirve para liberar el dispositivo.

11.4. CINTAS DE OTROS EQUIPOS

Cuando se trata de cintas de otros equipos no hay que inicializarlas y debemos montarlas con el calificador /FOREIGN. Si la cinta no fué grabada en código ASCII será necesario correr algún programa que convierta de EBCDIC a ASCII, pero es necesario conocer las características con que fue grabada la cinta.

11.5. Ejemplos:

Para copiar archivos a la cinta

```
$ ALLOCATE MTA0:
$ INIT/DENSITY=1600 MTA0: CINTA   cinta nueva, omítase
                                  si la cinta ya estaba
                                  inicializada
$ MOUNT     MTA0:
$ COPY MIARCHIVO.*  MTA0:[]*.*    copia todos los archivos
                                  que se llamen MIARCHIVO
$ DISMOUNT MTA0:
$ DEALL    MTA0:
```

Para copiar más archivos a una cinta ya inicializada

```
$ ALLOCATE MTA0:
$ MOUNT    MTA0:
$ SET DEF MTA0:
$ DIR                             directorio de la cinta
$ COPY DBA1:[CURSO]ELARCHIVO.*     *.*
```

```
                                        copia todos los archivos
                                        que se llaman ELARCHIVO
$ DISMOUNT MTA0:
$ DEALL    MTA0:

Para bajar de cinta a disco

$ ALLOCATE MTA0:
$ MOUNT    MTA0:
$ SET DEF MTA0:
$ DIR                                   directorio de la cinta
$ SET DEF DBA1:[CURSO]
$ COPY MTA0:MIARCHIVO.*     *.*         copia a disco todos los
                                        archivos de la cinta
                                        que se llamen MIARCHIVO
$ DISMOUNT MTA0:
$ DEALL    MTA0:
```
