C A N D E (Comandos y Ediciones) Del 11 al 17 de Mayo de 1983 .

Del 11 al 17 de Mayo	INTRODUCCION CONCEPTO Y CARACTERISTICAS DE USAR TIEMPO COMPARTIDO ESTRUCTURA DE LA B7800 Y CANDE \ COMANDOS DE CREACION COMANDOS DE EDICION COMANDOS DE CONSULTA COMANDOS DE EJECUCION USO DE PAQUETES MATEMATICOS Y ESTADISTICOS INTERCONEXION DE PROGRAMAS Y ARCHIVOS DE DATOS MEDIANTE WFL (WORK FLOW-LANGUAJE).	Miércoles a Martes de 18 a 21 h Sábado de 9 a 14 h	Ing.Armando Reyes González

EVALUACION DEL PERSONAL DOCENTE

CURSO: CANDE (COMANDOS Y EDICIONES)

FECHA: Del 11 al 17 de Mayo de 1983.

CONFERENCISTA	DOMINIO DEL TEMA	EFICIENCIA EN EL USO DE AYUDAS AUDIOVISUALES	MANTENIMIENTO DEL INTERES.(COMUNICACION CON LOS ASISTENTES, AMENIDAD, FACILIDAD DE EXPRESION).	PUNTUALIDAD	
1. ING. ARMANDO REYES GONZALEZ					
2.					
3.					
4.					
5.					
6.					
7.					
8.					
9.					

ESCALA DE EVALUACION : 1 a 10

EVALUACION DE LA ENSEÑANZA

SU EVALUACION SINCERA NOS AYUDARA A MEJORAR LOS PROGRAMAS POSTERIORES QUE DISEÑAREMOS PARA USTED.

TEMA	ORGANIZACION Y DESARROLLO DEL TEMA	GRADO DE PROFUNDIDAD LOGRADO EN EL TEMA	GRADO DE ACTUALIZACION LOGRADO EN EL TEMA	UTILIDAD PRACTICA DEL TEMA	
Introducción					
Concepto y Características de Usar Tiempo Compartido					
Estructura de la B7800 y CANDE					
Comandos de Creación					
Comandos de Edición					
Comandos de Consulta					
Comandos de Ejecución					
Uso de Paquetes Matemáticos y Estadísticos.					
Interconexión de Programas y Archivos de Datos Mediante WFL (Work Flow-Lanquaje).					

ESCALA DE EVALUACION: 1 a 10

EVALUACION DEL CURSO

	CONCEPTO	EVALUACION
1.	APLICACION INMEDIATA DE LOS CONCEPTOS EXPUESTOS	
2.	CLARIDAD CON QUE SE EXPUSIERON LOS TEMAS	
3.	GRADO DE ACTUALIZACION LOGRADO CON EL CURSO	
4.	CUMPLIMIENTO DE LOS OBJETIVOS DEL CURSO	
5.	CONTINUIDAD EN LOS TEMAS DEL CURSO	
6.	CALIDAD DE LAS NOTAS DEL CURSO	
7.	GRADO DE MOTIVACION LOGRADO CON EL CURSO	

ESCALA DE EVALUACION DE 1 A 10

1. ¿Qué le pareció el ambiente en la División de Educación Continua?

MUY AGRADABLE	AGRADABLE	DESAGRADABLE

2. Medio de comunicación por el que se enteró del curso:

PERIODICO EXCELSIOR ANUNCIO TITULADO DI VISION DE EDUCACIÑ CONTINUA	PERIODICO NOVEDADES ANUNCIO TITULADO DI VISION DE EDUCACIÑ CONTINUA	FOLLETO DEL CURSO

CARTEL MENSUAL	RADIO UNIVERSIDAD	COMUNICACION CARTA, TELEFONO, VERBAL, ETC.

REVISTAS TECNICAS	FOLLETO ANUAL	CARTELERA UNAM "LOS UNIVERSITARIOS HOY"	GACETA UNAM

3. Medio de transporte utilizado para venir al Palacio de Minería:

AUTOMOVIL PARTICULAR	METRO	OTRO MEDIO

4. ¿Qué cambios haría usted en el programa para tratar de perfeccionar el curso?

5. ¿Recomendaría el curso a otras personas?

SI	NO

6. ¿Qué cursos le gustaría que ofreciera la División de Educación Continua?

7. La coordinación académica fue:

EXCELENTE	BUENA	REGULAR	MALA

8. Si está interesado en tomar algún curso intensivo ¿Cuál es el horario más conveniente para usted?

LUNES A VIERNES DE 9 A 13 H. Y DE 14 A 18 H. (CON COMIDAS)	LUNES A VIERNES DE 17 A 21 H.	LUNES, MIERCOLES Y VIERNES DE 18 A 21 H.	MARTES Y JUEVES DE 18 A 21 H.

VIERNES DE 17 A 21 H. SABADOS DE 9 A 14 H.	VIERNES DE 17 A 21 H. SABADOS DE 9 A 13 Y DE 14 a 18 H.	O T R O

9. ¿Qué servicios adicionales desearía que tuviese la División de Educación Continua, para los asistentes?

10. Otras sugerencias:

C A N D E

C A N D E

MAYO, 1983

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

DR. OCTAVIO RIVERO SERRANO
Rector

LIC. RAUL BEJAR NAVARRO
Secretario General

C.P. RODOLFO COETO MOTA
Secretario General Administrativo

DR. JORGE HERNANDEZ HERNANDEZ
Secretario de Rectoría

LIC. IGNACIO CARRILLO PRIETO
Abogado General

ING. JORGE GIL MENDIETA
Director General del Programa
Universitario de Cómputo

C A N D E

PROGRAMA UNIVERSITARIO DE COMPUTO
FORMACION DE RECURSOS HUMANOS
ING. ARMANDO REYES G.
1982

C A N D E

INTRODUCCION

Una computadora de alta velocidad puede efectuar un número fantástico de cálculos por segundo. Esta rapidez ha hecho posible efectuar tareas que eran totalmente imprácticas en el pasado, lo cual implica que una computadora procese decenas de miles de "corridas" en un sólo día, quizá más de 100,000.

Puesto que el sistema de cómputo debe efectuar miles de tareas por día, debe de estar equipado con un programa ejecutivo para manejar sus tareas a velocidades que se miden en fracciones de segundo. Este programa decidirá que problemas deben ser corridos, cuanto tiempo se asignará a cada uno, responderá a las órdenes del usuario, llevar y traer información, asignar áreas de trabajo, etc. En síntesis administrará los recursos disponibles de la máquina y efectuará otro gran número de funciones.

En el párrafo anterior mencionamos que el sistema "responde a las órdenes del usuario", es decir, el usuario puede comunicarse directamente con la computadora a través de una terminal.

Y puesto que la máquina trabaja con tiempos medibles en microsegundos (10^{-12}) ésta "comparte" así sus recursos, dándole a cada usuario una pequeña atención cuando la necesita y proporcionándole respuestas casi inmediatas, además de que lo provee de recursos para almacenar sus programas.

Otra de las características importantes de la terminal es el uso interactivo con la computadora, por ejemplo, en programas que requieren la participación del usuario. Este tiene la impresión de tener el uso exclusivo de la máquina.

En resumen, encontramos que los sistemas de "tiempo compartido" proveen:

1.- Un servicio rápido.
2.- Terminales individuales para los usuarios y comunicación directa con la computadora.
3.- Interacción con los programas en ejecución.
4.- Facilidad de corregir y almacenar programas.

El lenguaje que se usa para trabajar a través de terminales mediante el sistema de tiempo compartido en las computadoras Burroughs series 6000 y 7000 es CANDE.

C A N D E

CANDE significa Command (órdenes o comandos) And (y) Edit (edición), por lo tanto, no ejecuta nada en sí, sino que sólo es el intermediario entre el usuario y el sistema operativo.

Comunmente se conoce a CANDE como el editor de la máquina Burroughs, ya que éste permite crear, editar y corregir programas o archivos de datos en una gran variedad de formas, y transferirlos de un dispositivo de almacenamiento a otro, tales como impresoras, discos y paquetes de discos (disk pack), cintas magnéticas y perforadoras de tarjetas.

Los programas creados a través de CANDE pueden ser ejecutados inmediatamente; mientras están corriendo, el usuario puede preguntar por ciertas características y el estado de su corrida. Sin embargo, también puede programar corridas vía proceso en lote ("Batch"), independientemente de la sesión, a través de la interfase CANDE con WFL (Work Flow Language, lenguaje de flujo de trabajo). Este lenguaje es empleado para controlar trabajos en "Batch", al igual que CANDE, es usado para controlarlos en "tiempo compartido".

Con CANDE, inclusive, es posible enviar mensajes a otras terminales, declarar ciertas características a la terminal, probarla o transferir el control de la terminal a un programa.

Estas notas no abarcan en su totalidad todos y cada uno de los comandos de CANDE, tampoco explican detalladamente las opciones posibles, sino que fueron creadas bajo los puntos de vista siguientes:

a) Comandos u órdenes de usuario y comandos de control más usados en CANDE.
b) Aquellas órdenes que se necesitan al empezar a emplear la computadora, a través de tiempo compartido, ya sea por vez primera o para usarla de nuevo.
c) Explicación breve y clara de los términos utilizados al trabajar con CANDE.
d) Orientación sobre el uso de los "diagramas de ferrocarril".

Las notas se dividen en tres partes:

I. Generalidades
II. Ordenes ó comandos de usuario
III. Ordenes ó comandos de control.

1. Generalidades

Al empezar a trabajar mediante CANDE, quizá el usuario se sorprenda cuando escuche en una conversación lo siguiente: "Al estar corriendo mi programa y preguntar por su estado el sistema se cayó y me saco fuera de sesión; espero poder recuperarlo". Ello significa que se estaba usando una terminal que al ejecutar un programa (correrlo) se estaba preguntando mediante un comando de control la situación o estado del mismo, en el sistema surgió una falla (se cayo) y dejo sin comunicación al usuario (lo saco de sesión) y al querer recuperarlo se espera que la información perdida sea mínima. Debido a la circunstancia antes descrita, es imprescindible que el usuario conozca la siguiente terminología.

- Terminal

Una terminal es un dispositivo con el cual es posible comunicarse directamente con la computadora. Puede ser terminal de teletipo, llamada a veces terminal, teleimpresora o terminal de teclado/impresor; consta de un teclado similar al de una máquina de escribir de oficina, también posee un dispositivo de impresión por el cual avanza un rollo de papel (identico a una máquina de escribir común y corriente) que produce como salida una "copia dura" (registro impreso) de la entrada, la información del sistema y los resultados del programa. Existe también una terminal llamada Unidad de Representación Visual (URV), que usa un tubo de rayos catódicos (TRC), parecida a una pantalla de televisión con teclado; la información es exhibida mucho más rápidamente que en una terminal teclado/impresor convencional (teletipo) y es silenciosa en su operación.

- Conectar

Significa establecer contacto físico con la máquina y dependerá de su terminal y su instalación particular; pero en general se dice que hay comunicación cuando al apretar la tecla RETURN (NEW LINE O ENTER, según sea el caso) la máquina responde imprimiendo información en su terminal.

Las terminales pueden ser conectadas de 2 formas:

a) Localmente.- Por medio de líneas privadas directas

b) Remotas.- Por línea telegráfica o telefónica.

Los terminos "local" y "remoto" se refieren al modo de enlace entre una terminal y la computadora; la distancia es relativa.

- Sesión

Una sesión es el período que transcurre entre el primer contacto con la computadora y la señal de salida. El usuario podrá realizar todo el trabajo que desee durante la sesión.

- Archivo

Un archivo es una sucesión de registros que contienen información, y en CANDE se clasifican en:

a) archivo de trabajo (WORKFILE)

b) archivo de biblioteca (LIBRARY FILE);

las que a su vez pueden ser de dos tipos:

a) que contengan información (datos), y

b) los que contengan instrucciones en lenguaje de programación de alto nivel.

Dentro de los primeros están aquellos que pueden ser:

SEQ (Secuencia de datos)

DATA (Datos manipulados en forma de palabras)

CSEQ (Secuencial de caracteres)

CDATA (Datos manipulados en forma de caracteres)

Dentro de los archivos de trabajo, los lenguajes reconocidos son: ALGOL, BASIC, COBOL, FORTRAN, PL/1, y aquellos que son únicos para los sistemas Burroughs, tales como el NDL (usado con el Net Work Definition Language) y JOB (usado con el Work Flow Language).

- Nombre de Archivos

El nombre de un archivo consiste en uno o más "identificadores", los cuales pueden contener hasta 17 caracteres y si se colocan entre comillas pueden contener caracteres especiales, números o espacios en blanco.

Ejemplos de nombres de archivos:

PROGRAMA

DATOS

2S

"*ESPECIAL*"

Cada archivo tiene un nombre, el cual puede estar asociado con otros nombres de archivos a través del uso de directorios, mismos que nos sirven para hacer referencia unicamente a un conjunto o subconjunto específico de datos y se construyen mediante el uso de la diagonal o "slash" Los directorios nos ayudan a lograr cierta organización de nuestra información, cada diagonal representa un nivel.

Al primer nivel se le llama directorio maestro y sólo se permiten 10 niveles de directorios.

Ejemplo de directorios:

PROGRAMAS/COBOL/NOMINA
PROGRAMAS/COBOL/INVENTARIOS
PROGRAMAS/BASIC/ALUMNOS
PROGRAMAS/BASIC/MNEMONICOS
PROGRAMAS/ALGOL/SISTEMA
PROGRAMAS/FORTRAN/SIMULACION
PROGRAMAS/FORTRAN/SIMULACION/1
PROGRAMAS/FORTRAN/SIMULACION/2
PROGRAMAS/FORTRAN/SIMULACION/3

- Archivo de Biblioteca

Cuando un archivo de trabajo es protegido mediante el comando SAVE, pasa a ser un archivo de biblioteca para el usuario, cuyo contenido no se pu-de alterar a través de CANDE, sino hasta que se encuentre como archivo de trabajo.

Al contrario de un archivo de trabajo, el usuario puede tener muchos archivos de biblioteca, pero se recomienda sólo mantener aquellos que seguirán usándose y destruir los que no se necesiten; esto es para beneficio suyo y de los demás usuarios, ya que así agiliza al sistema.

- Diagrama de Ferrocarril.

Un diagrama de ferrocarril es una técnica de representación gráfica que demuestra la sintaxis del empleo de las órdenes o comandos.

Por ejemplo, si se tuviese el siguiente diagrama de una órden o comando:

UNA ORDEN

«consiste de» «instrucciones»
«opciones»

El diagrama se leo de izquierda a derecha o en la dirección que nos indiquen las flechas.

Las mayúsculas deberán ser tecleadas tal y como se indican y las minúsculas que están entre corchetes angulares serán proporcionadas por el usuario. Lo subrayado indica que el comando permite abreviación. Así tendríamos que serán órdenes reconocidas por CANDE.

UNAORD
UNAORDEN
UNAORDEN <consiste de> <instrucciones>
UNAORDEN <consiste de> <opciones
UNAORDEN <consiste de> <instrucciones>, <opciones>
UNAORDEN <consiste de> <instrucciones>, <opciones><opciones>

- Reglas de uso de los diagramas de ferrocarril.

Las reglas básicas son:

1.- Los diagramas generalmente se leen de izquierda a derecha o en la dirección que las flechas nos indiquen.

2.- Las letras mayúsculas son usadas literalmente o pueden ser abreviadas si éstas están subrayadas (deben usarse tantos caracteres como se encuentren subrayados, no menos).

3.- Las letras minúsculas dentro de los corchetes angulares (< >) son variables, definidas por el usuario (los corchetes no forman parte de la instrucción).

4.- Los caracteres especiales, tales como * : , ! , . # + - / $? se emplean literalmente, excepto el símbolo de interrogación, usado también para las órdenes de control que pueden ser diferentes para cada estación.

5.- Por lo menos un espacio en blanco debe separar palabras adyacentes, pero no son necesarios en entrada alfanúmerica (palabras o números) y caracteres especiales.

6.- El símbolo 1 significa que la opción siguiente puede ser usada solamente una vez.
El entero indica el número de veces que puede ser usada la opción y no debe ser incluida en la orden.

7.- El símbolo 1* significa que la opción debe ser empleada por lo menos una vez.
El entero indica el máximo de veces que puede ser empleado.
Al igual que la regla anterior el entero y el asterisco no deben aparecer en la orden.

8.- Los símbolos A,B,C, que aparecen en algunos diagramas significan que la sintaxis de la orden continúa en el siguiente espacio disponible, debido a que el papel no es lo suficientemente ancho para colocar el diagrama entero, es decir, son conectores.

9.- El diagrama siempre finaliza ya sea con una barra ( ) ó con un diamante (.).

10.- Cuando el comando finaliza con una barra, significa que a continuación se puede emplear otro con el sólo hecho de usar punto y coma(;).
Se puede emplear tantos comandos como lo permita la línea física de la terminal. Ejemplo: LIST; GET PRUEBA; C; SA;E que con comandos cuyos diagramas terminan en barra.

11.- Cuando el comando finaliza con un diamante, significa que sólo se puede usar éste en turno, es decir, uno por línea.

- Opciones comunes.

Las siguientes opciones aparecen en varios comandos:

base — El primer número de línea que deberá ser usado en una serie de números de línea. (Esto es importante para nuestros archivos).

delim — Los delimitadores son todos aquellos caracteres no alfanuméricos (como el / * , . ¢ $ a # ;) . El caracter que se utilice para abrir es el que debe de usar para cerrar.

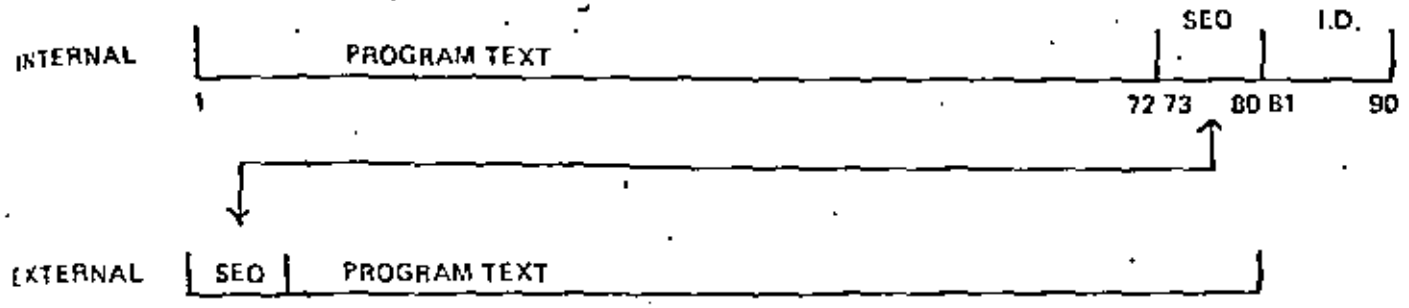
family name — Nombre de la familia del pack o disco.

inc — Incremento usado en los números de línea.

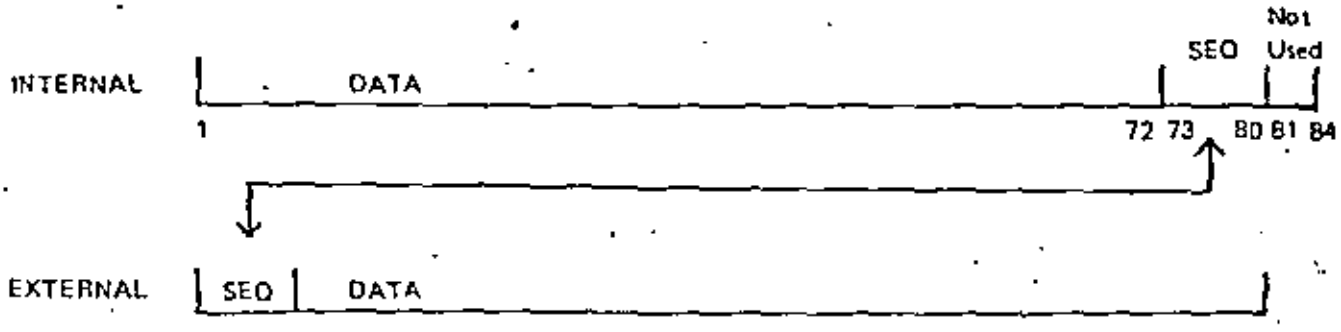
lsn — Número lógico de su estación.

mix no — Número de mezcla asignado por el sistema operativo para una "tarea", el cual será un dígito de 4 números.

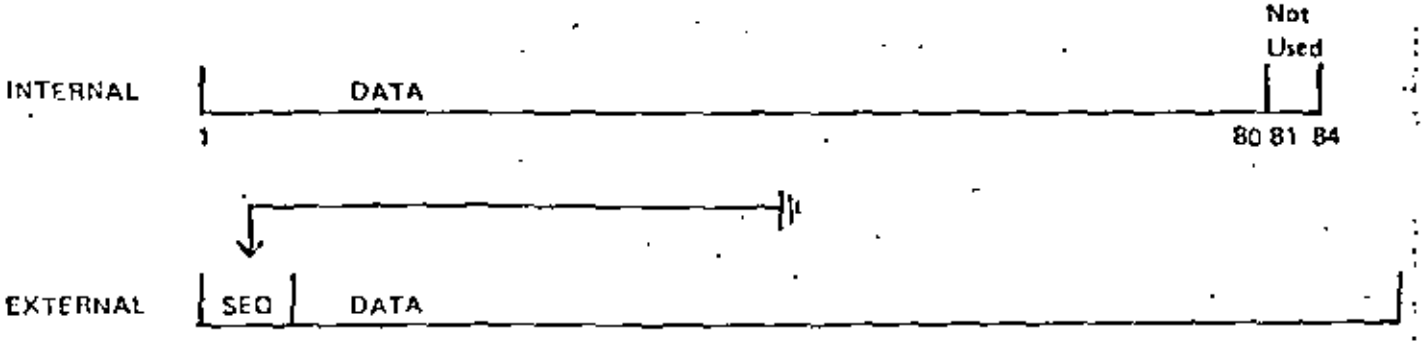
<sequence range list>	Un rango de secuencia del archivo
<file name>	nombre del archivo
<workfile>	archivo de trabajo
<directory name>	nombre del directorio
<source filename>	archivo fuente (programa fuente).
<object filename>	archivo objeto (programa objeto).
<depth>	profundidad (niveles en un directorio).
<column>	columna de nuestro registro
<target>	texto
<new text>	nuevo texto
<new filename>	nuevo nombre de archivo
<new directory name>	directorio
<recovery number>	número de recobro
<count>	cantidad (número entero)
<usercode>	clave
<password>	contraseña
<type>	tipo



Representation of type
ALGOL, BINDER, PL Files



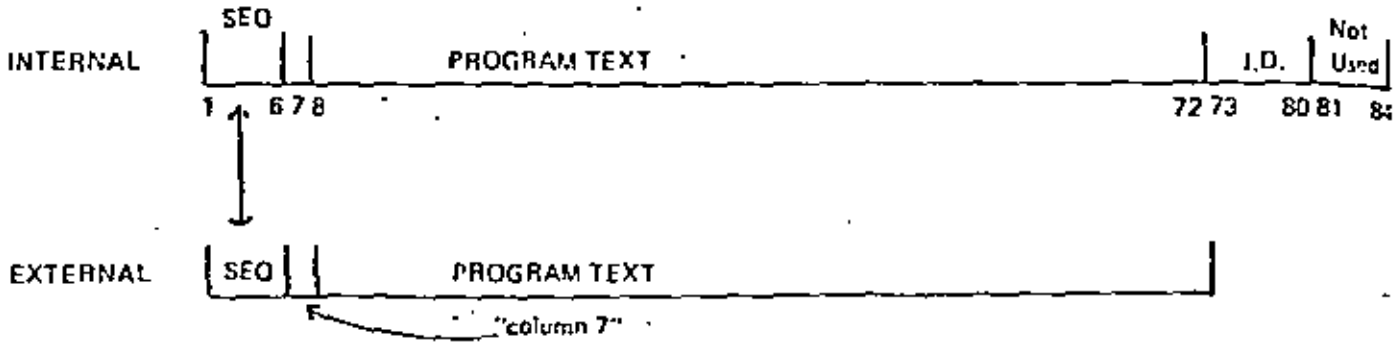
Representation of type
SEQ Files



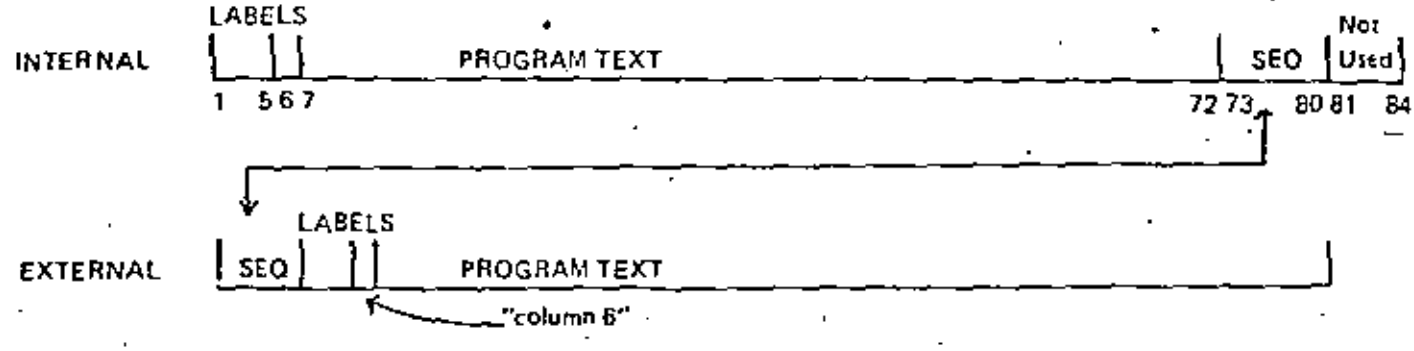
Representation of type
DATA Files

INTERNAL = Layout as stored in the computer.
EXTERNAL = Appearance of the record as the user sees it on the terminal.

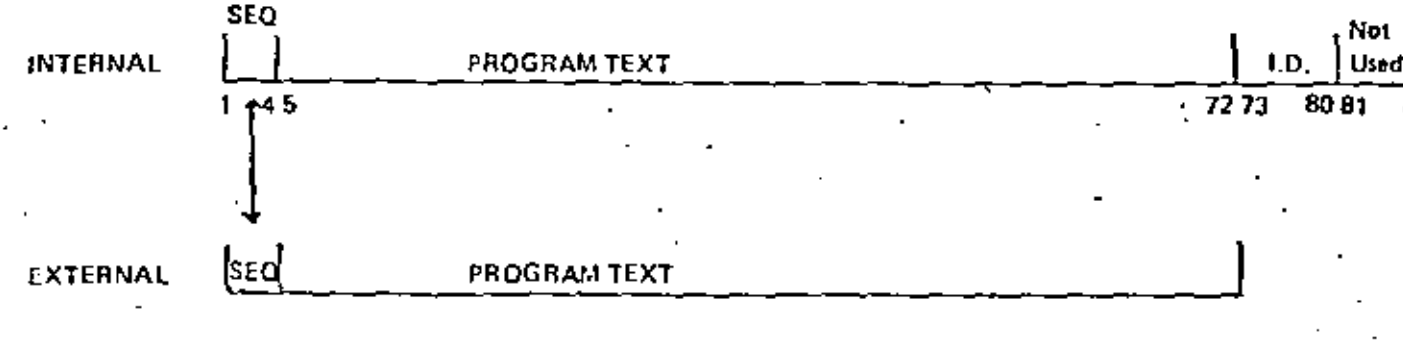
Figure 2.1. Representation of Data as Determined by File Type (Continued)



Representation of type
COBOL Files



Representation of type
FORTRAN Files



Representation of type
BASIC Files

INTERNAL = Layout as stored in the computer.
EXTERNAL = Appearance of the record as the user sees it on the terminal.

Figure 2.1. Representation of Data as Determined by File Type

II. Ordenes o comandos de usuario

Los diagramas no se escribirán ya que el usuario los puede consultar (en su tarjeta conocida también como "acordeón") o en el manual, por lo que el propósito de esta parte se limita solo a explicar el uso de los comandos mas comunes y las respuestas que deberá esperar del sistema.

HELLO Empieza una nueva sesión

Es el primer comando que se debe usar, una vez que se ha conectado. Para hacer uso de CANDE el usuario debe de tener una clave (USERCODE) de investigador, la cual consta de dos letras y un par de dígitos, un slash (diagonal) y una contraseña (password)'

Ejemplo:

HELLO
#B6700 CANDE 31.08; YOU ARE USUACU02(54)
ENTER USERCODE PLEASE
AZ90/PW
SESSION 5201 10:23:17 10/25/81

Donde todo lo subrayado es lo que el usuario escribe en la terminal y lo que empieza con el símbolo # será la respuesta de la máquina.

HELLO Primer comando que el usuario debe de usar para conectarse con la computadora y así entrar a una sesión de trabajo.

Símbolo con que responde el sistema

B6700 Máquina y serie con la que trabaja

CANDE Editor

31.08: Versión del sistema operativo

YOU ARE USUACU02 Nombre de su terminal

(54) Número lógico de su estación < lsn >

ENTER USERCODE PLEASE Solicita su identificación, o sea su clave.

AZ90/PW Esta es un ejemplo de clave y es otorgada en la sección de Relaciones del Centro de Servicios de Cómputo (CSC).

SESSION Esta conectado (en sesión) para que haga uso de la computadora.

5201 Número de mezcla asignado por el sistema mix no

10:23:17 Hora de entrada

10/25/81 Fecha

Cada vez que termine un comando o desee transmitir información presione la tecla RETURN (NEW LINE o ENTER, según sea el caso).

MAKE

Crea un archivo de trabajo, ya sea de datos ó para lenguaje de programación. Ejemplos:

MAKE DATOS
WORKFILE DATOS:SEQ

Crea un archivo de datos en orden secuencial (por omisión así será si no se especifica el tipo) llamado DATOS.

M PROGRAMA FORTRAN
WORKFILE PROGRAMA: FORTRAN

Crea un archivo de instrucciones en lenguaje FORTRAN llamado PROGRAMA.

El nombre del archivo de trabajo no debe ser igual al de un archivo de biblioteca, ya que entonces la orden será rechazada.

El usuario puede tener un archivo bajo un directorio empleando la diagonal. Ejemplo:

MAKE DIRECTORIO/PROGRAMA/ARCHIVO1FORTRAN
WORKFILE DIRECTORIO/PROGRAMAS/ARCHIVO1:FORTRAN

El Programa ha quedado en el directorio llamado DIRECTORIO. Bajo el directorio llamado PROGRAMA y el nombre del programa será ARCHIVO1. Este concepto de directorio es útil cuando usan la clave varios usuarios o se quiere hacer una identificación rápida de la información con que se cuenta.

SEQ

Con este comando, el usuario ya está en posición de comenzar a dar la información de su archivo de trabajo y para ello necesita de un número de línea, el cual será o no almacenado con el archivo de acuerdo al tipo que haya sido declarado en el MAKE. SEQ genera estos números de línea automáticamente para que vaya introduciendo su información

Ejemplo:

M CATEGORIAS ALGOL
WORKFILE CATEGORIAS : ALGOL
S
100

Se hace un archivo de trabajo llamado CATEGORIAS en lenguaje ALGOL e inmediatamente se da la secuencia del programa, es decir, el número de línea que nos servirá para identificar la información contenida en nuestro archivo.

Como sólo se dio la S, el número inicial o «base» (S1) será el 100, con incrementos «inc» de 100.

Ejemplo:

SEQ 500 + 50

500 BEGIN

550 FILE SALIDA (KIND=PRINTER).

600

Nos da una secuencia a partir del 500 y se irá incrementando de 50 en 50.

Ejemplos de opciones:

S 700 Inicia la secuencia en 700.

S NEXT Continúa en la secuencia en que se detuvo por algún motivo

S END Sigue inmediatamente después del último número de línea empleado

S END + 10 Sigue inmediatamente después del último número de línea, con un incremento de 10.

COMPILE

Una vez que su archivo de trabajo ha sido terminado, puede compilar su programa, que significa crear el programa objeto (no debe de ser datos, ya que éstos no se compilan sino que sólo se almacenan). Ejemplos:

C

COMPILING

ET=13.7 PT=1.0 IO=1.7

donde:

ET=13.7 Elapsed Time (tiempo de espera en la cola)

PT=1.0 Process Time (tiempo de proceso)

IO=1.7 IO Time (tiempos de Entrada/Salida)

Cuando la máquina nos ha dado esos tres tiempos, hemos de entender que el programa en sí no contiene errores de sintaxis; o bien antes de darnoslos proyecta en pantalla los errores que contenga el programa. Dado este caso el usuario procederá a la corrección de la sintaxis.

C PROGRAMA/ALMACENADO compila un archivo de biblioteca

C PROGRAMA/ALGOL WITH PASCAL compila programas en Pascal

C SINTAX revisa únicamente la sintaxis sin generar el progrma objeto.

FIX

Cuando un archivo de trabajo contiene errores o se desea modificar información, se empleará el comando FIX para su corrección o modificación.

Ejemplo:

F 750/BIGIN/BEGIN

donde:

F	Orden o comando
750	Número de línea
/	Delimitador
BIGIN	Texto viejo
/	Delimitador que cierra
BEGIN	Texto nuevo

Se modofica la cadena o letrero BIGIN por BEGIN.

F 800 7/?* Se modifica el primer slash por un asterisco.

* = . KID,DIND Se modifica la última línea donde se efectuo el cambio.

FIX 525 45-65 AFTER 0 C 0 OPIA Modifica la línea 525 a partir de la columna 45, y en la primera ocurrencia de la letra C, se insertará la cadena OPIA, quedando COPIA.

Si se usa la opción BEFORE quedará insertada la cadena OPIA antes de la letra C, leyéndose OPIAC.

Existe la opción INCL <delim> <target 1> <delim> <target 2> <delim> < new text > que sustituye desde <target1> hasta <target 2> con el < new text >

La opción EXCL es lo contrario de la opción anterior, respeta la información insertando el < new text > entre < target 1 > y < target 2 > ejemplo:

Supongamos que se tiene la línea

175 CENTRO DE SERVICIOS DE COMPUTO

y se usa

* 175 EXCL/CENTRO//DE//UNAM

la línea quedará

175 CENTRO UNAM DE SERVICIOS DE COMPUTO

Si se emplea la opción contraria,

* 175 INCL/CENTRO//DE//UNAM

la línea quedará:

175 UNAM SERVICIOS DE COMPUTO

líneas insertadas manualmente en su archivo de trabajo.

El usuario puede introducir líneas manualmente en su archivo de trabajo y el sistema lo incluirá automáticamente dentro de su archivo.
Si se quiere borrar una línea solo se debe teclear el número de ésta y apretar la tecla RETURN. Precausión: cuando altere una línea que no era la deseada, ya sea por falta de precausión o porque se empleó incorrectamente un comando; no podrá recuperarla si ya presionó la tecla RETURN; tendrá que volver a escribirla.

DELETE

Borrar líneas de una en una quizá no sea práctico por lo que existe el comando DELETE para eliminar líneas de su archivo de trabajo.

Ejemplos:

DELETE	40	Borra la línea 40
DEL	100- 700	Borra todas las líneas entre la 100 y la 700 incluyendo la 100 y la 700
DEL	10,25,75,90,5	Borra exclusivamente las líneas 10,25,75,90,5.
DEL	ALL	Borra todo el archivo de trabajo.

RUN

Ejecuta o corre un programa

RUN		Ejecuta su archivo de trabajo
R	PROGRAMA	Ejecuta su archivo de biblioteca llamado PROGRAMA
EXECUTE		Es sinónimo de RUN.

SAVE

Esta orden salva un archivo de trabajo, almacenándolo como un archivo de biblioteca.

Ejemplo:

SA	Salva su archivo de trabajo con el nombre con el que fue creado o sustituyéndolo por la versión antigua si fue llamado con GET o LOAD. Si se emplearon los comandos COMPILE, RUN o EXECUTE (y no hubo errores de sintaxis) se almacenan 2 versiones, o sea el programa fuente y el programa objeto; en archivos de datos sólo se almacena el fuente.
SA AS OTRO/NOMBRE	Se almacena su archivo de trabajo bajo el nombre OTRO/NOMBRE, respetando la versión antigua si este fue cargado con GET o LOAD; en cambio, si existe ya bajo ese nombre un archivo de biblioteca, el comando será rechazado.

SA SOURCE	Salva únicamente su programa fuente.
SA OBJECT	Salva únicamente su programa objeto.

REMOVE

Este comando remueve su archivo de trabajo, un archivo de biblioteca o todos los archivos bajo un directorio.

Ejemplos:

REM	Remueve su archivo de trabajo
REM SOURCE	Remueve únicamente el programa fuente respetando el objeto
REM OBJECT	Remueve el programa objeto respetando el programa fuente
REM NOMINA	Remueve un archivo de biblioteca llamado NOMINA.
REM PROG/COBOL	Remueve, tanto el archivo fuente como el objeto de un archivo de biblioteca.
REMOVE PROGRAMAS/BASIC/-	Remueve los archivos de biblioteca que se encuentran almacenados bajo el directorio PROGRAMAS/BASIC
REMOVE OBJECT/*	Remueve todos los programas objetos.

FILES

Esta instrucción produce un listado de los archivos de su biblioteca, de otro usuario o del sistema. (Dependerá mucho del tipo de seguridad con que fueron almacenados éstos).

FILES	Muestra todos los archivos de biblioteca e incluso el archivo de trabajo, si es que está presente.
FILES DIRECTORIO	Muestra unicamente aquellos archivos que se encuentran bajo el directorio llamado DIRECTORIO.
FILES:2	Muestra los archivos que están bajo un directorio con profundidad unicamente de dos niveles. Por ejemplo:

(A290)
. DIRECTORIO
. .PROGRAMAS

2 niveles de profundidad

LFILES — Este comando lista los atributos de sus archivos de trabajo

LFILES ALUMNOS/ — Lista los atributos de sus archivos que estén bajo el directorio ALUMNOS

LFILES CURSO: AREAS BLOCKSIZE CREA MAX MIN
Lista los atributos AREAS, BLOCK SIZE, CREATIONDATE, MAXRECSIZE, Y MINRECSIZE, de un archivo de biblioteca llamado CURSO.

LFILE:PRINTER — Lista los atributos por impresora.

GET

Carga o da una copia de un archivo de biblioteca, y la presenta como archivo de trabajo.
Ejemplos:

GET NOMINA — Carga el archivo de biblioteca como su archivo de trabajo

G NOMINA AS NOMINA2 — Carga archivo de biblioteca como archivo de trabajo llamado NOMINA2

G NOMINA 300-900 AS NOMINA3
Carga el archivo de biblioteca NOMINA de las líneas 300 a la 900, como archivo de trabajo cuyo nombre será NOMINA3.

LOAD

Es sinónimo de GET

LIST

Edita el contenido, línea por línea de algún archivo en su terminal.

L — Lista todo el archivo de trabajo

L PROGRAMA — Lista todo el archivo de biblioteca bajo el nombre de PROGRAMA.

L 1-600 — Lista unicamente las secuencias de la 1 a la 600.

L a 14-25 — Lista el contenido de su archivo solamente de las columnas 14 a la 25.

L : A — Muestra todas aquellas líneas que han sido alteradas en su archivo de trabajo antes de la última actualización (UPDATE).
Las líneas borradas son mostradas con un signo menos que precede al número de líneas; a las modificadas les antecede una F, a las insertadas una I y a las que fueron reemplazadas una R.

L : AF — Es idéntica a la opción A, excepto que muestra las líneas modificadas y la original en donde se hizo el cambio.

L : AFN	Combina el efecto de las opciones AF y AN.
L : AN	Muestra lo mismo que la opción A, excepto que en las inserciones muestra la línea anterior y posterior de la inserción.
L : ANF	Se combina el efecto de las opciones AF y AN.
L : CHANGES	Idéntico a la opción AF.
L : SQUASH	Elimina blancos múltiples, sustituyéndolos por uno solo.
L : UN	Lista el archivo sin secuencia.

WRITE

Manda el listado de un archivo de trabajo a biblioteca hacia impresora, unicamente vierte información, no indica errores de sintaxis.

W	Imprime su archivo de trabajo
W EXAMENES	Imprime un archivo de biblioteca llamado EXAMENES.
W 100-3000	Manda a impresión sólo la secuencia comprendida entre la 100 y la 3000.
W O 1	Imprime una sola columna de su archivo pero con número de línea.

W PROGRAMA/INVENTARIO : D, UN, S.	Manda a impresión un archivo de biblioteca que pertenece al directorio programa, lo escribirá a doble espacio, sin secuencia y compactando más de 2 espacios en uno solo.
W O 7 - 72 300-4250	Imprime de su archivo de trabajo las columnas 7 a la 72 del rango de secuencia entre el 300 y 4250 con el número de línea pegado a la derecha de la hoja de impresión.

PRINT es sinónimo de LIST.

FIND

Busca un texto dentro de un archivo

FIND/FILE/	Busca en el archivo todas las palabras que sean, empiecen, estén dentro de una cadena o finalicen con el letrero FILE.

La máquina nos responderá con una secuencia de números separada por comas, que son los números de línea en donde aparece la cadena FILE.

FIND LIST ¿ßHORAS&? Busca exactamente la cadena ßHORAS&; sólo aquellas palabras HORA, que tengan un blanco a la izquierda y a la derecha serán mostradas.

FIND /WRITE/ ARCHIVO7 950-10500 Busca la cadena WRITE dentro del archivo ARCHIVO7, en el rango 950-10500.

FIND .CASE. 14-20 : T Muestra la línea completa en donde aparezca la cadena CASE a partir de la columna 14 a 20. La opción T muestra la información que contiene la línea.

FIND/INTEGER/, # REAL #, .ARRAY. : T,S Busca las cadenas INTEGER, REAL Y ARRAY. Pueden por lo tanto hacerse varias búsquedas mediante un solo FIND.

FIND 5 /BEGIN/ ,. END. Busca los primeros cinco BEGIN y los cinco primeros END presentes en el archivo de trabajo.

FIND "PROCEDURE"PROGRAMA
FILE BUSQUEDA Busca todas las líneas en donde aparezca PROCEDURE, en el archivo de trabajo llamado PROGRAMA y almacena una copia de todas las líneas en un archivo de biblioteca llamado BUSQUEDA, que tendrá las mismas características y atributos que el archivo PROGRAMA.

REPLACE

Esta instrucción reemplaza o inserta textos en posiciones específicas dentro del archivo de trabajo. Puede inclusive borrar textos. Ejemplos:

REP/PRINT//WRITE/ Sustituye todas las cadenas PRINT por WRITE.

REP /TEXTO//VIEJO/ 600-700 Reemplaza TEXTO por VIEJO desde la línea 600 hasta la 700.

REP /LIN//REN/0
17-19 77-190

Reemplaza LIN por REN entre el rango 77 al 190 y de las columnas 17 a la 19.

REP COL 14 # ALGO #

Inserta a partir de la columna 14 la cadena ALGO en todo el archivo de trabajo.

REP.VIEJO..NUEVO.:T

Va reemplazando la información VIEJO por NUEVO y mediante la opción:T nos muestra cómo ha hecho la sustitución de la cadena de caracteres.

REP 17 /THY//THE/ 18-27 FILL REEMPLAZO 300-END

Los primeros 17 cambios que encuentre en el rango del archivo de trabajo de la 300 al último registro serán almacenados en un archivo llamado REEMPLAZO, el cual tendrá las mismas características que el archivo original, pero además serán los 17 primeros cambios que estén entre las columnas 18 y 27.

REP COL7/#RFC-/ : T, SQUASHED, TRUNCATED.

Inserta la cadena #RFC - a partir de la columna 7 sin borrar el contenido de la misma, muestra los cambios gracias a la opción :T, suprime blancos innecesarios y en caso de que al incluir la cadena en un registro éste se pase de la capacidad máxima (RECORD OVERFLOW) lo truncará.

Si no usa la opción TRUNCATED y existe el error anterior la máquina nos responderá:

#SKIPPED

Indicándonos que no se hizo el reemplazo en esa línea, sino que fue saltada.

REP / % / # COMMENT # 9-60

Sustituye la palabra COMMENT en donde se encuentra el % pero a partir de la columna 9 a la 60 los % que esten antes de la columna 9 ó después de la 60 serán respetados.

RESEQ

Esta instrucción asigna nueva secuencia en los números de línea de su archivo de trabajo.

Ejemplos:

RES	Da una nueva secuencia de 100 en 100
RES 100-300	Sólo altera esa parte del archivo
RES 100-300 10 + 10	Altera el rango 100 al 300, modificándole como secuencia de 10 en 10.
RES 100	Cambia el orden de la secuencia de los números de línea de 100 en 100.
RES OVER	Da un número de línea a un archivo que no lo necesita almacenar (principalmente del tipo DATA).

Nota:

Cuando se usa RES en un archivo en lenguaje BASIC, el sistema verifica que las direcciones de las transferencias de control condicionada e incondicionada concuerden con la nueva secuencia (no es válido para GOSUB).

INSERT

Copia líneas del archivo de trabajo que se tenga en terminal o de un archivo de biblioteca al archivo de trabajo.

Ejemplos:

IN 700 AT 900	Copia la línea 700 y le asigna el número de línea 900.
IN 400 - 800 ALMACENADO AT END	Copia las líneas de la 400 a la 800 del archivo de biblioteca de nombre ALMACENADO al final del archivo de trabajo que se tenga en terminal.
IN 900 - 1500 AT 10 + 5	Copia la información del rango de líneas 900 a 1500 y la coloca a partir de la posición 10 con un incremento de 5 en 5.

MOVE

Transfiere unas líneas de su archivo de trabajo a otra área, asignándoles nuevo número.

Ejemplos:

MO 707-810 TO 10	Mueve el rango de líneas 707-810 a la posición 10 con incremento de 100.
MO 14 TO 900	Transfiere la línea de la posición 14 a la 900.
MO 95 TO NEXT	Transfiere la secuencia 95 al siguiente número de línea, después del último movimiento que se hizo asumiendo el incremento con que se realizó éste.

Nota:

En caso de que se inserten o muevan líneas y éstas se traslapen con números de línea ya existentes, los movimientos no serán llevados a cabo, o sea que el comando será rechazado.

RECOVER

Recobra un archivo de trabajo que fue protegido cuando el sistema sufrió una falla. Este archivo presenta siempre un número, mediante éste se puede recobrar o remover.

Cuando se emplea REC sin parámetros, la máquina nos muestra los archivos de recobro.

Ejemplo:

```
REC
# RECOVERY DATA ON PACK
296 PRUEBAS (04/19/82)
270 NOMINA (04/19/82)
280 ETIQUETAS (09/19/82)
```

Donde:

290 es el número de recobro

PRUEBAS nombre del archivo

(09/19/82) Fecha de protección

Para recuperar alguno de estos archivos se debe hacer mención del <número de recobro> Ejemplo:

```
REC 290
, la máquina no responderá
# WORKFILE PRUEBAS:  FORTRAN 90 RECORDS, SAVED
```

Automáticamente el archivo de recobro pasa a ser archivo de trabajo; pero cuando en la terminal existe un archivo de trabajo, el comando sera rechazado.

Se puede proteger un archivo si se combinan el comando SAVE y el RECOVER.

Ejemplo:

GET PRUEBAS

WORKFILE PRUEBAS: FORTRAN 90 RECORDS, SAVED

SAVE RECOVER

#

RECOVER

RECOVERY DATA ON PACK

290 PRUEBAS (04/19/82)

270 NOMINA (09/19/82)

280 ETIQUETAS (04/19/82)

291 PRUEBAS (09/20/82)

DISCARD

Lo contrario a recobrar un archivo, usted puede descartar archivos de recobro.

Ejemplos:

DIS 270, 280, 290 Remueve 3 archivos de recobro.

TITLE

Cambia el nombre de un archivo de trabajo, de archivos de biblioteca o de un directorio.

Ejemplos:

TITLE CAMBIA	Modifica el nombre original del archivo de trabajo con que fue creado (MAKE) o cargado (GET o LOAD) por el nombre CAMBIA.
TITLE CAMBIOS TO ALTAS	Modifica el nombre del archivo de biblioteca llamado CAMBIOS, dejándole el título ALTAS.
TITLE ARCHIVOS/*TO PROGRAMAS/*	Cambia el nombre del directorio o programas, la máquina indica el número de archivos cuyo título fue modificado.
TITLE PROGRAMA/ FORTRAN/SEIS TO BIBLIOTECA/FORTRAN/ CORRECTO	Cambia un sólo programa del directorio PROGRAMA/FORTRAN llamado PROGRAMA/FORTRAN/CORRECTO respetando todos los demás programas bajo los mismos directorios.

UPDATE

Actualiza el archivo con las modificaciones que se le hayan hecho.

Notas:

Este comando trabaja automáticamente si el usuario usa cualquiera de las siguientes ordenes.

INSERT, MOVE, RESEQ, COMPILE, EXECUTE, RUN, FIND, REPLACE o SAVE.

SO

Muestra y enciende las opciones de mensajes de la terminal que el usuario esté empleando.

WHAT

Muestra el nombre y la situación de un archivo de trabajo. Esta información dependerá del estado en que se encuentre el archivo de trabajo, y puede ser: nombre y tipo del archivo de trabajo, número de líneas; si existe un archivo objeto y si éste y el fuente están protegidos..

ADD Y COPY

Las instrucciones ADD y COPY nos permiten transferir archivos de un dispositivo de almacenamiento a otro.

Para ilustrar se dará como ejemplo la transferencia de un archivo de disco a cinta y viceversa.

COPY = FROM CINTA (UNITNO =88)

Los subrayado indica que deben ir en la instrucción y lo que no está, es variable, es decir, CINTA será el nombre que usted le dé a su cinta de trabajo y el 88 será la unidad que el operador (SPO) le asigne.

La instrucción copia el contenido de la cinta a disco.

COPY ARCHIVO FROM CINTA (UNITNO=88)

Copia un sólo archivo de la cinta a disco.

COPY ARCHIVO AS NUEVO FROM CINTA (UNITNO=88)

Copia un sólo archivo, bajo un nuevo nombre.

COPY ARCHIVO1, ARCHIVO2, ARCHIVON FROM CINTA (UNITNO=88)

Copia varios archivos

COPY ARCHIVO1AS NUEVO1, ARCHIVO2 AS NUEVO2, ARCHIVO1 AS NUEVO1, ARCHIVON AS NUEVON FROM CINTA (UNITNO=88)

Copia varios archivos, cambiándoles el nombre.

COPY PROGRAMAS/* FROM CINTA (UNITNO=88)

Copia todos los archivos que estén bajo el directorio PROGRAMAS, exclusivamente.

COPY PROGRAMAS/*,ARCHIVOS/*, DATOS/* FROM CINTA

Copia todos los archivos de los directorios PROGRAMAS, ARCHIVOS y DATOS.

COPY * TO CINTA (UNITNO=88)

Para copiar todos los archivos de disco a cinta.

COPY ARCHIVO TO CINTA (UNITNO=88)

Queda grabado un sólo archivo en la cinta.

COPY VIEJO AS NUEVO FROM CINTA (UNITNO=88)

Añade un archivo de cintas a disco cambiando el nombre

COPY VIEJO AS NUEVO TO CINTA (UNITNO=88)

Copia un solo archivo de disco a cinta cambiandole el nombre.

COPY ARCHIVO1, ARCHIVO2, ARCHIVO1, ARCHIVON TO CINTA (UNITNO=88)

Copia varios archivos a la cinta.

COPY ARCHIVO1 AS NUEVO1, ARCHIVO2 AS NUEVO2, ARCHIVO3, AS NUEVO3 TO CINTA (UNITNO=88)

Copia los archivos en disco a la cinta con nuevos nombres.

COPY FAMILIA/* TO CINTA (UNITNO=88)

Copia todos los archivos bajo el directorio FAMILIA a cinta.

COPY FAMILIA1/-, FAMILIA2/-, FAMILIA3/* TO CINTA
(UNITNO-88)

Copia los archivos de los directorios FAMILIA1, FAMILIA2 y FAMILIA3, respetando todos los demás directorios en el disco.

Observaciones:

ADD tiene el mismo efecto que el COPY, con la diferencia de que si un archivo ya existe en el dispositivo a donde se quiere transferir, el archivo no será copiado, que es el caso contrario del COPY.

En los ejemplos anteriores, donde dice COPY sustituir por ADD, si emplea la opción 6 COMPARE, el proceso es lento y emplea mucho tiempo de procesado, sólo úsela cuando la información sea muy valiosa.

Cuando usted desee bajar información de cinta a disco, éste debe ir sin aro (S/A), pero si quiere grabar sobre ella, deberá pedirla con aro (C/A). Precausión: Evite confusiones y cerciórese de lo que realmente desea hacer, ya que cada vez que graba en la cinta magnética, ésta es purgada y la información ya no podrá ser recuperada.

III. Ordenes o Comandos de Control.

Existen algunas órdenes que se ejecutan antes que cualquier otra, es decir, tienen prioridad, a estas se les conoce como COMANDOS DE CONTROL.
Mediante ellos puede ganar un control más inmediato, comprobar sus colas de comandos, verificar los programas que están corriendo, preguntar sobre la situación en que se encuentran, etc. Estos empiezan con un caracter especial tal como el signo de interrogación (?), que se le llama en su estación como "caracter de control".

Sin embargo, al emplearlo usted sólo podrá usar uno y sólo uno de estos, ya que si se fija, los diagramas de su sintaxis terminan con un diamante, estos son muy útiles cuando esta efectuando corridas o controlando eventos en los que pueden ocurrir errores. Los comandos de control más comunes son:

?C
Lista las tareas terminadas. La información por cada tarea completa incluye:

Número del trabajo (JOB)
Número de mezcla (MIX)
Motivo de la terminación (EOJ, EOT, DSED, SNTX, etc)
Nombre de la tarea
Nombre del Compilador

Ejemplos:

```
?C
----- NO  COMPLETED  ENTRIES -----
```

No ha habido entradas.

```
?C
----- COMPLETED  ENTRIES -----
* 4086/4112 0-DS (A290) OBJECT/EJEMPLO ON PACK
* 4086/4109 EOT   CANDE WRITER
* 4086/4100 EOT   FORTRAN OBJECT/EJEMPLO
```

Muestra 3 tareas:

*4086/4112 Es la más reciente o indica que ha sido abortada una tarea (O-DS)

*4086/4109 Se mando imprimir un archivo (WRITER)

*4086/4100 Terminó una tarea FORTRAN.

?COUNTS

Muestra el nivel corriente de actividad de CANDE. La respuesta incluye el número de estaciones "activas" (usuarios conectados ó archivos abiertos en la terminal); el número de estaciones "reconocidas" (terminales conectadas); el número de "tareas" (programas corridos a través de CANDE); y para cada proceso CANDE el cual haya manejado edición de texto y funciones de salida, el número de "trabajadores" (los procesadores logicos hacen estas funciones).

Ejemplo:

```
?COUNT
# 8 TASKS, 2 WORKERS 37 STATIONS ACTIVE,
65 ATTACHED.
```

? CS

Muestra el estado en que se encuentra una compilación. La respuesta incluye el número de línea que va analizando y el número de errores de Sintaxis encontrado. Si no se especifica algún número de mezcla (<mix no>) tomará la compilación de su archivo de trabajo.

Ejemplos:

```
?CS
SEQ - 00002000, NO ERRORES
No ha habido errores en la compilación
? 4579 CS
SEQ - 00001500, ERROR COUNT=2
Se encuentra en la línea 1500, del trabajo
cuyo <mix no> es el 4579 y lleva 2 erro-
res encontrados
?CS
# NOT BUSY
Termino la compilación
```

? DS

Termina un trabajo o tarea. Terminar un trabajo, se refiere'aquellos que han sido hechos mediante la interfase de MFL y las tareas aquellas que son hechas en la sesión de CANDE.

Ejemplos:

?DS

Termina con la tarea que esté corriendo en su terminal en ese momento.

?4597 DS

Termina con la tarea o trabajo que tenga ese número de mezcla específico.

?END

Este comando niega la petición de entrada desde su terminal. Cierra el archivo de entrada mandando la señal de Fin de Archivo (EOF) al archivo de Entrada abierto a través de terminal.

Nota: No es útil en programas BASIC ya que este no reconoce Fin de Archivo a través de INPUT.

?JA

Es útil para preguntar por trabajos (JOB) corridos a través de batch. La respuesta incluye:

Si esta en espera, catalogado (SCHEDULED) o situación de error.

Número del trabajo (JOB)

Número de mezcla

Prioridad

Nombre de la tarea

Nombre del compilador

?MIX

Muestra los trabajos que estén corriendo bajo su clave.

Por cada MIX activos la respuesta incluye:

catalogo (SCHEDULED), en espera ó situación de error

número del trabajo

número de mezcla (MIX)

prioridad

nombre del trabajo (JOB)

mensaje RSVP (mensaje que requiere respuesta del usuario u operador)

mensaje exhibido

?MSG

Muestra los mensajes más recientes asociados a sus tareas.

Ejemplo:

?MSG

-----MESSAGES-----

Ejemplo:

```
?MSG
----- MESSAGES -----
  * 4396 OPERATOR DSED   20D: 041A:4*
ha sido abortada una tarea
```

?OK

Reactiva el procesamiento de una tarea suspendida. Se puede especificar una tarea especifica con su número de mezcla.

?SQ

Muestra los trabajos en espera de ser corridos. Puede preguntar por una cola específica (clase), dando el número de esta, después del comando, la información que se muestre será el número de mezcla asociado al trabajo y la posición que ocupa en la cola.

Ejemplos:

```
?SQ 5
QUEVE 5:
    3201  JOB  TRABAJO (#0004)
    3246  JOB  ESTRUCTURA (#0015)

?SQ
QUEVE 1:
NO ENTRIES
QUEVE 2:
    NO ENTRIES
QUEVE 3:
    4300 JOB URGENTES (#004)
QUEVE 4:
    NO ENTRIES
QUEVE 5:
    3201 JOB TRABAJO (#004)
    3246 JOB ESTRUCTURA (#0015)
QUEVE 66:
    NO ENTRIES
```

?SS

Manda un mensaje de una terminal a otra o al operador del sistema, es decir:

?SS <lsn> mensaje que desee transmitir

?SS <SPO> mensaje a la consola del operador

En caso de que la terminal no este encendida, el mensaje no llegará a su destino.

?ST

Detiene temporalmente una tarea. Si no especifica número de mezcla toma la que este corriendo a través de su terminal.

Ejemplo:

```
?ST
# 6150 OPERATOR STOPPED
? 6140 ST
#6140 OPERATOR STOPPED
```

NOTA: Se reactiva con ?OK ,aborta con ?DS

?STATUS

Muestra la situación ó estado de su terminal ó de los comandos CANDE ó tareas.

Ejemplos:

```
?STA
#12:15 SEQ= 00073000  ET=6.6 PT=4.1 IO=1.9
```

Va en la línea 00073000 y muestra los tiempos empleados.

```
?STA
#12:25 NOT BUSY
Terminó la tarea
```

?TI

Muestra los tiempos usados por el sistema para una tarea se puede preguntar por una tarea específica mediante el número de mezcla. Los tiempos mostrados son de proceso (CP), de entrada/salida (IO) y de espera (ET).

Ejemplo:

```
?TI
   TIMES FOR 6150
     PROCESS = 00:00 :25
     IO      = 00:01 :30
     ELASPSED= 00:00 :05
```

? TIME

Muestra la hora, fecha y día de la semana.

Ejemplo:

```
?TIME
# 17:10 PM FRIDAY, JUNE 25, 1982
```

?TO

Manda un mensaje a un usuario. Si hay varios usuarios conectados a la misma clave, a todos y cada uno de ellos le llega éste.

Ejemplo:

```
?TO  AZ90  <mensaje>
```

?WD

Muestra la fecha

?WHERE

Pregunta por una clave en especial, si está o no conectado al sistema vía CANDE, la información que nos dá es el usuario por el que se preguntó, el nombre de la estación en donde se encuentra y el número lógico de la estación.

?WHY

Pregunta por el estado de una tarea que esté corriendo, si no se especifica con el número de mezcla < mix no. > se toma por omisión el que esté activo en su terminal al momento de preguntar. La respuesta del comando incluye: La hora, clase o cola, prioridad, origen (estación), el estado del stack, nombre de la tarea, nombre del compilador, último mensaje mostrado (si existe), último mensaje de respuesta.

?WRU

Muestra la identificación de su estación y el software CANDE i.e.

```
B6700 CANDE 30.08; YOU ARE
#SESSION = 4086 USER =AZ90
```

#SESSION = 4086 Número de mezcla de la sesión de CANDE

USER = AZ90 Identificación del usuario en esa sesión

USUACU(55) Identificación de la estación

?WT

Muestra la hora del día.

?Y

Sinónimo de WHY

Ejemplos:

```
?Y
    STATUS OF TASK 6209 AT 09:45:23
    PRIORITY = 50
    ORIGINATION: LSN 55
    STACK STATE: WAITING ON AN EVENT
    PROGRAMA NAME: (AZ90)/CAMBIOS ON UNAM3

?6209 Y
STATUS OF TASK 6209 AT 09:49:37
PRIORITY = 50
ORIGINATION :LSN 55
STACK STATE : ACTIVE
PROGRAMA NAME: (AZ90)/CAMBIOS ON UNAM3.
```

BIBLIOGRAFIA

1. B6700/B6600 SERIES
   CANDE
   USER'S MANUAL
   NO. 5001712
   1977 BURROUGHS CORPORATION

2. INTRODUCCION AL SISTEMA DE TIEMPO COMPARTIDO VIA CANDE
   TECNINOTAS
   CENTRO DE SERVICIOS DE COMPUTO (UNAM)
   CARMEN BRAVO DE CARIO Y EUMELIA MENDOZA DE BASAVE
   80/2 JULIO, 1980

3. USO DE LAS INSTRUCCIONES "COPY" Y "ADD" DE W.F.L.
   EN LA B6700
   TECNINOTAS
   CENTRO DE SERVICIOS DE COMPUTO (UNAM)
   MARGARITA RAMOS ELORDUY
   78/11 OCTUBRE, 1978.

C A N D E

USO DE LENGUAJE CANDE

MAYO. 1983

INTRODUCCION

ESTE TRABAJO FUE EDITADO POR MEDIO DE TIPOGRAFO AUTOMATIZADO (COMUNMENTE CONOCIDO COMO SERVICIO/TIA), DEL DEPTO. DE ATENCION AL USUARIO DEL P.U.C. DE LA U.N.A.M.

FUE ELABORADO POR:

ALDANA JAMAICA GUADALUPE ROSALIA.

CALDERON LARA AGUSTIN

CORTES GARCIA HECTOR PATRICIO

CRUZ CORRAL ALFONSO MANUEL

DOMINGUEZ PASTRANA LEONARDO

ESCAMILLA GARCIA JUAN CARLOS

MIEMBROS DEL PROGRAMA UNIVERSITARIO DE COMPUTO,
SUPERVISADO POR :

SILVIA LARRAZA HERNANDEZ

ARMANDO REYES GONZALEZ

MEXICO, D.F., VERANO/1982.

EL OBJETIVO DE ESTE TRABAJO ES SERVIR DE GUIA A LAS PERSONAS QUE TIENEN LA NECESIDAD DE USAR UNA COMPUTADORA BURROUGHS, DE LAS SERIES B6000/B7000 EN FORMA INTERACTIVA.

EL MODO NO-INTERACTIVO, PARA FINES DIDACTICOS, LO COMPONEN: LA LECTORA DE TARJETAS, EL PROCESADOR CENTRAL Y UNA IMPRESORA, ESTE MODO DE TRABAJO ES CONOCIDO COMO "BATCH", LA INFORMACION FLUYE A TRAVES DE LOS TRES ELEMENTOS CITADOS, PARA LOGRAR OBTENER RESULTADOS. EL PROGRAMA DEBE IR ACOMPAÑADO DE LOS DATOS NECESARIOS PARA SU EJECUCION CORRECTA.

VARIOS PROGRAMAS QUE YA HAN SIDO EJECUTADOS CORRECTAMENTE, CONSTITUYEN UNA "BIBLIOTECA" PARTICULAR , A CADA UNO DE ELLOS SE LE DA EL NOMBRE DE "ARCHIVO" ; ASI, EN UN MOMENTO DADO, SE TOMA DE LA BIBLIOTECA UN ARCHIVO, SE LE INTRODUCEN NUEVOS DATOS Y SE LLEVA A PROCESAR. A PARTIR DE QUE SE ENTREGA NO SE REQUIERE DE LA PARTICIPACION DEL USUARIO. EN CAMBIO EN EL MODO INTERACTIVO SE PUEDEN IMPLEMENTAR TRABAJOS DINAMICOS QUE REQUIEREN DE LA PARTICIPACION DE ESTE DURANTE EL PROCESAMIENTO , AQUI EL ARCHIVO NO ESTARA CONSTITUIDO DE TARJETAS SINO QUE SERA UN PROGRAMA GRABADO EN DISCO O CINTA MAGNETICA Y QUE HA SIDO CREADO CON ANTERIORIDAD.

EL MODO INTERACTIVO SOLO SE HACE POSIBLE CON EL USO DE TERMINALES QUE PUEDEN SER:

PANTALLAS O TUBO DE RAYOS CATODICOS (CTR'S).

Y

TERMINAL DE TELEIMPRESION O DE TECLADO-IMPRESOR.

PARA EL EMPLEO DE ESTOS DISPOSITIVOS SE CREO 'CANDE' (COMMAND AND EDIT), LENGUAJE CONSTITUIDO POR UNA SERIE DE COMANDOS QUE PERMITEN: CREAR, MODIFICAR, GUARDAR, ACCESAR, EJECUTAR, DESTRUIR, ETC., UN ARCHIVO.

EN 'CANDE', SE TIENEN DOS CLASES DE ARCHIVOS:

1.- DE TRABAJO (WORKFILE): ES UNO Y SOLO UNO, ES AQUEL CON EL QUE SE ESTA TRABAJANDO Y EN EL QUE PUEDEN EFECTUARSE MODIFICACIONES.

2.- DE BIBLIOTECA (LIBRARY FILES): ESTOS SE ENCUENTRAN ALMACENADOS EN DISCO, PUEDEN SER UNO O MAS, NO ES POSIBLE ALTERAR LA INFORMACION CONTENIDA EN ELLOS.

UN ARCHIVO DE TRABAJO ESTA 'PRESENTE', EL DE BIBLIOTECA NO LO ESTA, EN EL PRIMERO ES POSIBLE REALIZAR MODIFICACIONES, A LOS SEGUNDOS SOLO SE LES PUEDE CONSULTAR.

LOS ARCHIVOS PUEDEN SER DE LOS SIGUIENTES TIPOS :

ALGOL
BASIC
COBOL
DATA
FORTRAN
PL/1
SEQ
ETC.

PRIMERA SESION: QUE HAY QUE HACER? 5

HAY QUE LOCALIZAR EL INTERRUPTOR DE ENCENDIDO Y ACTIVAR LA TERMINAL (EN CASO DE QUE ESTE APAGADA). UNA VEZ ENCENDIDA SE TECLEA EL PRIMER COMANDO, ESTE ES 'HELLO', CON EL SE INDICA AL SISTEMA QUE UN USUARIO ESTA DISPUESTO A TRABAJAR EN MODO INTERACTIVO, ES DECIR, DESEA INICIAR UNA SESION.

INMEDIATAMENTE DESPUES SE OPRIME LA TECLA 'RETURN' O 'NEW LINE' O 'ENTER' SEGUN SEA EL CASO.

LA MAQUINA RESPONDE CON EL SIGNO '#' PIDIENDO CLAVE DE USUARIO (USERCODE) Y SU PALABRA DE PASO (PASSWORD).

EJEMPLOS

```
HELLO
#B7800:126 CANDE 31.280: YOU ARE DESPFI1(24)
#ENTER USERCODE PLEASE
SL82
#ENTER PASSWORD PLEASE.
XX
#DEFAULT PRINT DESTINATION=SITE
```

```
HELLO SL82/XX
#DEFAULT PRINT DESTINATION=SITE
```

```
**REMOCION DE ARCHIVOS NO ACCESADOS TECLEE NEWS**
*RECOVERY DATA ON PACK
  390 FINAL ((07/01/82)
*SESSION 8624  10:54:51  07/02/82
```

PARA TERMINAR LA SESION DEBE UNO DESPEDIRSE DE LA MAQUINA USANDO EL COMANDO "BYE" .

EJEMPLOS

```
BYE
*END SESSION 8996 ET=50.8 PT=0.0 IO=0.0
*USER = SL82 09:24:30 06/12/82
```



```
BYE
*END SESSION 5529  ET=12:11.0 PT=0.0 IO=0.0
*USER = SL82  14:33:57  06/15/82
```


EN RESUMEN: "HELLO" JUNTO CON LA CLAVE DE USUARIO (USERCODE) Y LA PALABRA DE PASO (PASSWORD), CONECTAN AL USUARIO CON EL SISTEMA Y EL COMANDO "BYE" LO DESCONECTA DEL SISTEMA.

ESTAS SE PRESENTAN CUANDO SE TIENE NECESIDAD DE CREAR, CORREGIR, EJECUTAR, DESTRUIR, ETC. UN ARCHIVO, LO CUAL SE LOGRA MEDIANTE LA AYUDA DE LOS SIGUIENTES COMANDOS:

MAKE

ESTE COMANDO SIRVE PARA CREAR UN ARCHIVO Y DEBE IR ACOMPAÑADO DE UN IDENTIFICADOR (NOMBRE DEL ARCHIVO) Y DE UN TIPO

MAKE <IDENTIFICADOR> <TIPO>

EL IDENTIFICADOR DEBE SER ORIGINAL, ES DECIR, NO DEBE EXISTIR OTRO ARCHIVO CON EL MISMO NOMBRE. EL TIPO PUEDE OMITIRSE; Y EN TAL CASO EL SISTEMA ASUMIRA QUE ES DE DATOS, TIPO SECUENCIAL (SEQ).

ESTE NO PODRA SER USADO SI SE TIENE PRESENTE ALGUN OTRO ARCHIVO DE TRABAJO.

EJEMPLOS

1.-OMITIENDO EL TIPO

```
MAKE PRUEBASCANDE
#WORKFILE PRUEBASCANDE: SEQ
SEQ
100 ESTE ES UN ARCHIVO CREADO COMO
200 MUESTRA DEL USO DEL COMANDO
300 MAKE HABIENDO OMITIDO EL TIPO
400 ES TODO
500
#
REMOVE
#
```

```
MAKE SINTIPO
#WORKFILE SINTIPO: SEQ
SEQ
100 ESTE CONJUNTO DE LINEAS FORMAN UN ARCHIVO
200 DEBIDO A LA OMISION DEL TIPO, EL SISTEMA
300 ASUMIO QUE SE TRATA DE UN ARCHIVO SECUENCIAL
400 ES TODO
500
#
SAVE
#UPDATING
#WORKSOURCE SINTIPO SAVED
#
```


2.-SIN OMISION DEL TIPO.

```
MAKE CONTIPO/1 PASCAL
#WORKFILE CONTIPO/1: PASCAL
SEQ
100 PROGRAM EJEMPLO(INPUT,OUTPUT);
200 VAR
300      INDICE : INTEGER;
400   BEGIN
500     WRITELN(' LOS NOMBRES DE LOS DIGITOS SON:');
600     FOR INDICE:=0 TO 9 DO
700        CASE INDICE OF
800           0: WRITELN('CERO',INDICE:5);
900           1: WRITELN('UNO',INDICE:5);
1000          2: WRITELN('DOS',INDICE:5);
1100          3: WRITELN('TRES',INDICE:5);
1200          4: WRITELN('CUATRO',INDICE:5);
1300          5: WRITELN('CINCO',INDICE:5);
1400          6: WRITELN('SEIS',INDICE:5);
1500          7: WRITELN('SIETE',INDICE:5);
1600          8: WRITELN('OCHO',INDICE:5);
1700          9: WRITELN('NUEVE',INDICE:5);
1800       END(*ENDCASE*);
1900 END(*ENDPROGRAM*).
```

```
MAKE MOVIMIENTOS DATA
#WORKFILE MOVIMIENTOS: DATA
SEQ
100 ESTE ARCHIVO ESTA HECHO CON EL OBJETO DE
200 REALIZAR MOVIMIENTOS CON SUS LINEAS
300 VEAMOS:
400 ESTA ES LA PRIMERA LINEA
500 ESTA ES LA SEGUNDA LINEA
600 OBSERVA LOS MOVIMIENTOS QUE SOBRE
700 TODO EL ARCHIVO SE REALIZARAN
800 LISTO, EMPEZAREMOS.
900
#
```



```
SAVE
```

ESTE COMANDO SIRVE PARA ALMACENAR UN ARCHIVO DE TRABAJO DENTRO DE LOS ARCHIVOS DE BIBLIOTECA.

ALGUNAS DE LAS OPCIONES VALIDAS, SON LAS SIGUIENTES:

1.- SAVE SOURCE :- SALVA SOLAMENTE LA VERSION FUENTE DE UN ARCHIVO DE TRABAJO

2.- SA OBJECT :- SALVA SOLAMENTE LA VERSION OBJETO DE UN ARCHIVO DE TRABAJO.

3.- SA RECOVERY :- ASIGNA A UN ARCHIVO DE TRABAJO CARACTERISTICAS QUE GENERAN UN ARCHIVO DE RECOBRO.

4.- SA AS < IDENTIFICADOR > :- SALVA EL ARCHIVO DE TRABAJO BAJO OTRO NOMBRE, DISTINTO DEL QUE TIENE.

5.- SA AS *< IDENTIFICADOR > :- SALVA SOLO LA VERSION FUENTE CON OTRO NOMBRE.

EJEMPLOS

```
SAVE
#WORKSOURCE HECANDE SAVED; OLD SOURCE REMOVED


SAVE AS EJEMP/CANDE/2
#WORKFILES CONTIPO/1 SAVED AS (SL82)EJEMP/CANDE/2


SAVE AS EJCANDE
#WORKSOURCE CANDEJ SAVED AS (SL82)EJCANDE ON PACK
#


SAVE
#WORKSOURCE CANDE1 SAVED; OLD SOURCE REMOVED
```



```
WHAT
#WORKFILE FIBONACCI: FORTRAN, 14 RECORDS, SAVED
#OBJECT FILE PRESENT, SAVED
SA REC
#
WHAT
#NO WORK FILE
```


REMOVE

ESTE COMANDO SIRVE PARA REMOVER (BORRAR) ARCHIVOS, QUE PUEDEN SER:

UN ARCHIVO DE TRABAJO.

UN ARCHIVO DE BIBLIOTECA.

UN DIRECTORIO DE ARCHIVOS DE BIBLIOTECA.

ALGUNAS DE LAS OPCIONES VALIDAS SON:

1.- REM SOURCE .

2.- REM OBJECT

LA ESPECIFICACION SOURCE BORRA SOLO LOS ARCHIVOS FUENTE; LA ESPECIFICACION OBJECT BORRA LOS ARCHIVOS OBJETO, EN CASO DE NO PONER ALGUNA DE ESTAS ESPECIFICACIONES, SERAN REMOVIDAS AMBAS VERSIONES.

3.- REM <IDENTIFICADOR>

4.- REM <NOMBRE DE DIRECTORIO>/=

CUANDO SE QUIERE REMOVER UN ARCHIVO DE BIBLIOTECA, ES NECESARIO INDICAR EL <IDENTIFICADOR>; SI ES UN DIRECTORIO, SE

```
INDICA <NOMBRE DE DIRECTORIO>/=.


    EJEMPLOS



     REM LASMOVIDAS
     # (ISCA)LASMOVIDAS ON IIMAS REMOVED


    FILES DIREC
    (SL82) ON PACK
    .  DIREC
    .  ;  1 ; ALGOL
    .  .  2 ; ALGOL
    .  .  3 ; ALGOL
    #
    REM DIREC/=
    # 3  FILES IN (SL82)DIREC/= REMOVED ON PACK
    # 1  FILE IN (SL82)OBJECT/DIREC/= REMOVED ON PACK
    #
```



```
REM SOURCE FIBONACCI
# (SL82)FIBONACCI ON PACK REMOVED



REMOVE OBJECT FIBONACCI
# (SL82)OBJECT/FIBONACCI ON PACK REMOVED


REMOVE PRUEBA
# (SL82)PRUEBA (1 OBJECT) ON PACK REMOVED
```

SEQ

ESTE COMANDO SIRVE PARA GENERAR UN NUMERO DE SECUENCIA AUTOMATICO PARA LAS LINEAS DEL TEXTO DE UN ARCHIVO DE TRABAJO. GENERALMENTE SE USA DESPUES DEL COMANDO MAKE.

SE PUEDEN EMPLEAR LAS OPCIONES:

1.- SEQ <BASE> :- DONDE BASE DEFINE EL NUMERO INICIAL DE LA SECUENCIA (QUE DEBE SER ENTERO POSITIVO).

2.- SEQ NEXT :- ASIGNA A LA PRIMERA LINEA EL NUMERO DE SECUENCIA SIGUIENTE AL USADO POR EL ULTIMO COMANDO SEQ, MOVE, INSERT O RESEQ. EN CASO DE NO HABER USADO NINGUNO DE ESTOS, EL VALOR QUE POR OMISION TENDRA LA PRIMERA LINEA SERA DE 100.

3.- SEQ END :- ASIGNA A LA PRIMERA LINEA EL ULTIMO NUMERO DE SECUENCIA DEL ARCHIVO MAS EL INCREMENTO CONSIDERADO.

EJEMPLOS

```
MAKE PRUEBASCANDE
#WORKFILE PRUEBASCANDE: SEQ
SEQ
```



```
100 ESTE ES UN ARCHIVO CREADO COMO
200 MUESTRA DEL USO DEL COMANDO
300 MAKE HABIENDO OMITIDO EL TIPO
400 ES TODO
500
#
```

```
MAKE SINTIPO
#WORKFILE SINTIPO: SEQ
SEQ
100 ESTE CONJUNTO DE LINEAS FORMAN UN ARCHIVO
200 DEBIDO A LA OMISION DEL TIPO, EL SISTEMA
300 ASUMIO QUE SE TRATA DE UN ARCHIVO SECUENCIAL
400 ES TODO
500
#
```

```
MAKE TAREA JOB
#WORKFILE TAREA: JOB
SEQ
100 RUN #SERVICIO/TIA/FILE FUENTE(TITLE=FINAL)
200 END
300
#
```

ESTOS SON: GET O LOAD, LIST O PRINT, FIX O *, DELETE, 23
COMPILE Y RUN O EXECUTE.

EN RESUMEN; SE SABE COMO LOGRAR UNA SESION, COMO CREAR UN ARCHIVO, COMO SALVARLO O GUARDARLO, COMO REMOVERLO O BORRARLO Y COMO DESPEDIRSE O TERMINAR UNA SESION.

EN ESTA PARTE SE EXPLICARA LO QUE SE DEBE HACER SI: SE DESEA TENER PRESENTE ALGUN ARCHIVO QUE YA SE CREO CON ANTERIORIDAD (USANDO GET O LOAD)) VER EL CONTENIDO DE DICHO ARCHIVO (USANDO LIST O PRINT)) CORREGIR ALGUNA LINEA O REGISTRO (USANDO FIX O *)) VERIFICAR QUE LA SINTAXIS 'REGLAS DE CONSTRUCCION DE LAS INSTRUCCIONES DE UN PROGRAMA' SEA CORRECTA (USANDO COMPILE)) FINALMENTE EJECUTARLO (USANDO RUN O EXECUTE)

'LOS COMANDOS ANTES MENCIONADOS SE DESCRIBIRAN EN LAS SIGUIENTES PAGINAS .

GET O LOAD

SON SINONIMOS Y SU FINALIDAD ES TRAER COMO ARCHIVO DE TRABAJO UNA COPIA DE UN ARCHIVO DE BIBLIOTECA, AL QUE SE TIENE LA NECESIDAD DE EDITAR (CORREGIR).

EL USO DE ESTOS DEPENDE DE LA SEGURIDAD QUE TENGA EL ARCHIVO DE BIBLIOTECA (EN EL CASO DE QUE ESTE NO PERTENEZCA A LA CLAVE DEL USUARIO)

ALGUNAS OPCIONES VALIDAS SON:

1.- GET <IDENTIFICADOR> !- DONDE EL IDENTIFICADOR INDICA EL ARCHIVO QUE SE DESEA ACCESAR.

2.- LOAD <IDENTIFICADOR> ON <NOMBRE DE FAMILIA> !- SI EL ARCHIVO QUE SE DESEA ACCESAR NO ESTA EN LA FAMILIA QUE SE ENCUENTRA TRABAJANDO, SIENDO NECESARIO ESPECIFICAR 'ON <NOMBRE DE LA FAMILIA>'.

3.- GET <IDENTIFICADOR> <RANGO> !- CUANDO SOLO SE DESEA ACCESAR UNA PARTE DEL ARCHIVO; DICHO RANGO PUEDE SER UNA, DOS O MAS LINEAS; SI SON MAS DE DOS SE DEBE INDICAR EL LIMITE INFERIOR Y EL LIMITE SUPERIOR SEPARADOS CON UN GUION.

4.- LOAD <IDENTIFICADOR> AS <IDENTIFICADOR>!- CUANDO SE

DESEE QUE EL ARCHIVO DE TRABAJO ADQUIERA UN NOMBRE DISTINTO AL DEL ARCHIVO DE BIBLIOTECA.

AMBOS COMANDOS, GET Y LOAD NO PODRAN SER USADOS SI SE TIENE PRESENTE ALGUN ARCHIVO DE TRABAJO.

EJEMPLOS

```
LOAD LASHOVIDAS
#WORKFILE LASHOVIDAS: DATA, 8 RECORDS, SAVED
#
```

```
0 CANDE1
#WORKFILE CANDE1: SEQ, 711 RECORDS, SAVED
```

```
LOAD CANDE1 100-10850 AS ACL1
#WORKFILE ACL1: SOURCE IS (SL82)CANDE1 ON PACK:SEQ
#
```

```
LOAD CANDE1 10900-20150 AS ACL2
#WORKFILE ACL2: SOURCE IS (SL82)CANDE1 ON PACK:SEQ
#
```



```
GET CANDE1 21050-21550 AS ACL
#WORKFILE ACL: SOURCE IS (SL82)CANDE1 ON PACK:SEQ
#
```

LIST O PRINT

SON SINONIMOS Y SU OBJETIVO ES MOSTRAR, TOTAL O PARCIALMENTE, EL CONTENIDO DE UN ARCHIVO EN TERMINAL. ESTE PUEDE SER: EL DE TRABAJO O DE BIBLIOTECA.

ENTRE ALGUNAS OPCIONES VALIDAS, TENEMOS:

1.- LIST <IDENTIFICADOR> :- PARA LISTAR UN ARCHIVO DE BIBLIOTECA, SE REQUIERE EL <IDENTIFICADOR> Y, DE SER POSIBLE, TAMBIEN 'ON<NOMBRE DE FAMILIA>', PARA HACER MAS FACIL EL ACCESO.

2.- LIST:SQUASHED :- PRESENTA UN LISTADO OMITIENDO LOS BLANCOS INNECESARIOS.

3.- PRINT:UNSEQUENCED :- MUESTRA UN LISTADO DEL ARCHIVO DESEADO, PERO SIN LOS NUMEROS DE SECUENCIA.

SI SOLO SE DESEA LISTAR PARCIALMENTE SE NECESITA ESPECIFICAR EL <RANGO> . CON LIST:A, LIST:AN, LIST:AF, LIST:AFN Y LIST:COMPARE, SE OBTIENEN LOS REGISTROS QUE FUERON MODIFICADOS, INSERTADOS O BORRADOS ANTES DE LA ULTIMA ACTUALIZACION.

EJEMPLOS

```
LIST
100 .ESTA ES LA PRIMERA LINEA
200  ESTA ES LA SEGUNDA LINEA
300  ESTE ARCHIVO ESTA HECHO CON EL OBJETO DE
400  REALIZAR MOVIMIENTOS CON SUS LINEAS
500  VEAMOS:
600  OBSERVA LOS MOVIMIENTOS QUE SOBRE
700  TODO EL ARCHIVO SE REALIZARAN
800  LISTO EMPEZAREMOS
*
```

```
PRINT
100  OBSERVA LOS MOVIMIENTOS QUE SOBRE
200  TODO EL ARCHIVO SE REALIZARAN
300  LISTO, EMPEZAREMOS.
400  ESTA ES LA PRIMERA  LINEA
500  ESTA ES LA SEGUNDA LINEA
600  ESTE ARCHIVO ESTA HECHO CON  EL OBJETO DE
700  REALIZAR MOVIMIENTOS CON SUS LINEAS
800  VEAMOS:
*
```

```
PRINT 100-END !U
OBSERVA LOS MOVIMIENTOS QUE SOBRE
TODO EL ARCHIVO SE REALIZARAN
LISTO, EMPEZAREMOS.
ESTA ES LA PRIMERA LINEA
ESTA ES LA SEGUNDA LINEA
```

```
ESTE ARCHIVO ESTA HECHO CON EL OBJETO DE
REALIZAR MOVIMIENTOS CON SUS LINEAS
VEAMOS:
*

WHAT
*WORKFILE MOVIBLE/SOURCE IS (ISCA)LASMOVIDAS
          ON LINAS: DATA, 8 RECORDS
LIST
100  OBSERVA LOS MOVIMIENTOS QUE SOBRE
200  TODO EL ARCHIVO SE REALIZARAN
300  LISTO, EMPEZAREMOS.
400  MIRA ESTA ERA LA PRIMERA LINEA
500  MIRA ESTA ERA LA SEGUNDA LINEA
600  ESTE ARCHIVO MIRA ESTA HECHO CON EL OBJETO DE
700  REALIZAR MOVIMIENTOS CON SUS LINEAS
800  VEAMOS:
*
```


FIX O *

SIRVE PARA MODIFICAR O CORREGIR UNA LINEA O REGISTRO DE UN ARCHIVO DE TRABAJO.

LA OPCION * SE USA EN LUGAR DE FIX, CUANDO SE DESEA CORREGIR UNA LINEA TENIENDOSE EL MODO AUTOMATICO DE SECUENCIA.

ALGUNAS DE LAS OPCIONES VALIDAS SON:

1.- FIX <NUMERO DE SECUENCIA> :-DONDE EL NUMERO DE SECUENCIA ES LA LINEA QUE SE PRETENDE MODIFICAR.

2.- FIX= :- SI YA SE EFECTUO UNA CORRECION EN UNA LINEA DETERMINADA E INMEDIATAMENTE SE DESEA CORREGIR OTRO ASPECTO EN LA MISMA ,YA NO ES NECESARIO INDICAR EL NUMERO DE SECUENCIA, SOLO BASTA AGREGAR EL SIGNO = .

EJEMPLOS

```
F1100.DE NUESTRA.LA
*
F=@ON@GN DEL USUARIO
*
```

```
L17100
17100 FAMILIA QUE SE ESTA TRABAJANDO,PUEDE NO USARSE
#
F17100.,PUEDE NO, DEBE
#
L*
17100 FAMILIA QUE SE ESTA TRABAJANDO DEBE USARSE
#
```


COMPILE

ESTE COMANDO TIENE LA FINALIDAD DE VERIFICAR LA SINTAXIS DE UN PROGRAMA Y, EN CASO DE RESULTAR CORRECTA, GENERA EL CODIGO CORRESPONDIENTE.

ALGUNAS DE LAS OPCIONES PERMITIDAS SON:

1.- COMPILE <IDENTIFICADOR> :- CUANDO NO ES EL ARCHIVO DE TRABAJO

2.- C <IDENTIFICADOR> ON <NOMBRE DE FAMILIA> :- SI EL ARCHIVO QUE SE DESEA COMPILAR NO SE ENCUENTRA EN LA FAMILIA QUE SE ESTA TRABAJANDO.

3.- C <IDENTIFICADOR> AS <IDENTIFICADOR OBJETO>

4.- C <IDENTIFICADOR> AS $<IDENTIFICADOR OBJETO>

SI SE DESEA QUE EL CODIGO QUE SE GENERE RECIBA UN NOMBRE ESPECIAL DENTRO DEL DIRECTORIO DE ARCHIVOS OBJETO, SE INDICA 'AS<IDENTIFICADOR OBJETO>'. SI SE DESEA SUPRIMIR EL PREFIJO OBJECT SE INDICA 'AS$<IDENTIFICADOR OBJETO>'.

5.- COMPILE SINTAX :- SI SOLO SE REQUIERE VERIFICAR LA SINTAXIS,PERO NO GENERAR CODIGO .

6.- C <NOMBRE DE COMPILADOR> :- SI EL TIPO DEL ARCHIVO FUENTE NO ES EL MISMO QUE EL LENGUAJE EN EL QUE ESTA ESCRITO, ES NECESARIO ESPECIFICAR <NOMBRE DEL COMPILADOR> COMO: ALGOL, BASIC, COBOL, DCALGOL, ESPOL, FORTRAN Y PL/I.

7.- COMPILE WITH <NOMBRE DE COMPILADOR> :- CUANDO EL COMPILADOR NO ES ESTANDAR.

EJEMPLOS

```
C W PASCAL
#UPDATING
#WAITING FOR AVAILABLE TASK
#QUEUED
?STA
#14:08 1ST IN TASK QUEUE
#COMPILING 5486
```



```
COMPILE WITH PASCAL
#UPDATING
#WAITING FOR AVAILABLE TASK
?STA
#14:42 3RD IN TASK QUEUE
```

```
C W PASCAL
#UPDATING
#WAITING FOR AVAILABLE TASK
?STA
#14:46 9TH IN TASK QUEUE
#COMPILING 1274
?CS
%EQ = 00000010, NO ERRORES
#ET=12.1 PT=2.1 IO=1.5
```

```
R
#UPDATING
#COMPILING 2421
#ET=7.6 PT=1.3 IO=1.5
```

RUN O EXECUTE

SON SINONIMOS Y SIRVEN PARA EJECUTAR (CORRER) UN PROGRAMA.

SE PUEDEN EMPLEAR LAS SIGUIENTES OPCIONES:

1.- EXECUTE $<IDENTIFICADOR> :- AL EMPLEARSE ESTA OPCION, SE SUPRIME EL PREFIJO 'OBJECT' DEL <IDENTIFICADOR> EN EL DIRECTORIO DE ARCHIVOS OBJETO.

2.- RUN <IDENTIFICADOR> :- SI SE DESEA EJECUTAR UN ARCHIVO DE BIBLIOTECA .

3.- EXECUTE <IDENTIFICADOR> ON <NOMBRE DE FAMILIA> :- SI EL ARCHIVO A EJECUTARSE NO ESTA EN LA FAMILIA EN LA CUAL EL USUARIO ESTA TRABAJANDO.

EJEMPLOS


```
RUN
#WAITING FOR AVAILABLE TASK .
?STA
#14:51 7TH IN TASK QUEUE
#RUNNING 1322
 LOS NOMBRES DE LOS DIGITOS SON:
CERO    0
UNO     1
TRES    3
CUATRO  4
CINCO   5 .
SEIS    6
SIETE   7
OCHO    8
NUEVE   9
#ET=11.3 PT=0.3 IO=0.9
```

```
RUN FIBONACCI
#UPDATING
#SCHD 1843
#COMPILING 1843
#ET=12.1 PT=1.1 IO=1.3
#WAITING FOR AVAILABLE TASK .
#RUNNING 1851
#ET=29.2 PT=0.3 IO=1.1
```

∂e

PARA CORREGIR UN ARCHIVO SE USO FIX, QUE SOLO PERMITE CORREGIR UN REGISTRO A LA VEZ. HABRA OCASIONES EN QUE SEA NECESARIO CORREGIR VARIOS REGISTROS SIMULTANEAMENTE (CON REPLACE) BORRAR ALGUN(OS) REGISTRO(S) (CON DELETE); CAMBIAR DE POSICION REGISTROS COMPLETOS (CON MOVE); HACER DE DOS ARCHIVOS UNO SOLO O DUPLICAR LINEA(S) (CON INSERT); O BIEN, EN CASO DE NO DISPONER YA DE ESPACIO PARA MAS INSERCIONES, DARLE NUEVOS NUMEROS DE SECUENCIA (CON RESEQ).

ESTOS COMANDOS SE CONOCEN CON EL NOMBRE DE COMANDOS DE ACTUALIZACION.

REPLACE

SIRVE PARA REEMPLAZAR UN TEXTO DETERMINADO DE UN ARCHIV DE TRABAJO POR OTRO TEXTO NUEVO.

LOS <DELIMITADORES> PUEDEN SER CUALQUIER CARACTER ESPE CIAL SIEMPRE Y CUANDO SEAN HOMOGENEOS. LA COMA NO DEBE UTILI ZARSE COMO DELIMITADOR.

OPCIONES VALIDAS :

1.- REPLACE LITERAL

CUANDO EN EL <TEXTO A REEMPLAZAR> HAY ESPACIOS EN BLAN CO, NO ES NECESARIO CONSIDERARLOS PUES NO SON SIGNIFICATIVOS SIN EMBARGO, SI SE DESEAN TOMARSE EN CUENTA, DEBE USARSE ESTA OPCION.

2.- REP FIRST

3.- REP <CANTIDAD>

LA OPCION 'FIRST' LIMITA EL REEMPLAZO A LA PRIMERA OCU RRENCIA DEL <TEXTO A REEMPLAZAR> EL CUAL APARECE DENTRO DE LA LINEA O REGISTRO, <CANTIDAD> TAMBIEN LIMITA EL NUMERO DE REM PLAZOS, Y <RANGO> RESTRINGE EL NUMERO DE REEMPLAZOS A UNA PARTE DETERMINADA DEL ARCHIVO DE TRABAJO.

4.- REP<DELIM><TEXTO><DELIM><DELIM><TEXTO><DELIM>:TEXT

5.- REP<DELIM><TEXTO><DELIM><DELIM><TEXTO><DELIM>:SEQUENCE

SI SE DESEA SABER SI EL REEMPLAZO FUE EFECTUADO SE DEBE USAR LA OPCION 'SEQUENCE' O, SI SE DESEA VERIFICAR EL CONTENIDO DEL REEMPLAZO SE DEBE USAR LA OPCION 'TEXT'.

SI EL CAMPO EN EL REGISTRO ES INSUFICIENTE PARA EL <NUEVO TEXTO> Y EXISTEN ESPACIOS INECESARIOS EN EL, AL UTILIZAR LA OPCION 'SQUEEZE' SE REDUCEN A UNO SOLO LOGRANDOSE EFECTUAR CORRECTAMENTE EL REEMPLAZO.

EJEMPLOS

```
FIND/ESTA/
#WORKFILE MOVIMIENTOS
400,500,600
#
REPLACE /ESTA/ /MIRA ESTA/
#WORKFILE MOVIMIENTOS
#UPDATING
#
LIST
100  OBSERVA LOS MOVIMIENTOS QUE SOBRE
200  TODO EL ARCHIVO SE REALIZARAN
300  LISTO, EMPEZAREMOS
400  MIRA ESTA ES LA PRIMERA LINEA
500  MIRA ESTA ES LA SEGUNDA LINEA
600  ESTE ARCHIVO MIRA ESTA HECHO CON EL OBJETO DE
700  REALIZAR MOVIMIENTOS CON SUS LINEAS
800  VEAMOS
#
```

```
REPLACE/ES//ERA/IT
$WORKFILE MOVIMIENTOS
$UPDATING
400 MIRA ESTA ERA LA PRIMERA LINEA
500 MIRA ESTA ERA LA SEGUNDA LINEA
$
```


DELETE

ES UTIL PARA SUPRIMIR LINEAS DEL TEXTO DE UN ARCHIVO DE TRABAJO.

ES PERMITIDO EMPLEAR LAS SIGUIENTES OPCIONES:

1.- DEL <RANGO>

2.- DEL ALL

SE UTILIZA <RANGO> PARA INDICAR CUALES SON LAS LINEAS QUE DEBERAN SER SUPRIMIDAS, PUEDE SER UNA, DOS O MAS. SI SON MAS DE DOS Y ESTAN EN SECUENCIA CONTINUA, SOLO BASTA INDICAR LOS LIMITES INFERIOR Y SUPERIOR SEPARADOS POR UN GUION, EN CASO CONTRARIO SE SEPARAN MEDIANTE COMAS.

CUANDO SE UTILIZA LA OPCION 'ALL' ES EQUIVALENTE A USAR EL COMANDO MAKE, YA QUE PURGA EL ARCHIVO (ELIMINA TODA LA INFORMACION CONTENIDA EN EL).

EJEMPLOS

```
WHAT
#WORKFILE CAMBIO/1 1 SEQ
LIST
 100 ARCHIVO1 LINEA1
 200 ARCHIVO1 LINEA2
 300 ARCHIVO1 LINEA3
#
DELETE 100
#


LIST
 200 ARCHIVO1 LINEA2
 300 ARCHIVO1 LINEA3
#


DELETE ALL
#
PRINT
#
```


MOVE

SIRVE PARA CAMBIAR LINEAS DENTRO DE UN ARCHIVO DE TRABAJO, MOVIENDOLAS DE UN LUGAR A OTRO.

ALGUNAS OPCIONES VALIDAS SON:

1.- MOVE <RANGO> TO <RANGO>

2.- MOVE <RANGO> TO END

3.- MOVE <RANGO> TO <BASE>+<INCREMENTO>

EL <RANGO> INDICA LA(S) LINEA(S) QUE SE DESEA(N) MOVER; SI EL MOVIMIENTO ES HACIA EL FINAL SE EMPLEA LA CLAUSULA 'TO END'.

SI SE USA LA OPCION <BASE>+<INCREMENTO> SE DEBE TENER CUIDADO QUE NO SE TRASLAPEN LOS NUMEROS DE SECUENCIA PORQUE DE SER ASI, NO SE EJECUTARA EL COMANDO.

EJEMPLOS

```
LIST
500 ARCHIVO2 LINEA1
600 ARCHIVO2 LINEA2
700 ARCHIVO2 LINEA3
800 ARCHIVO2 LINEA4
900 ARCHIVO2 LINEA5
1000 ARCHIVO 3
1100 ARCHIVO 3
1200 ARCHIVO 3
1300 ARCHIVO 1 LINEA 1
1400 ARCHIVO 1 LINEA 2
#
MOVE 1300-END TO 100+50
#UPDATING
#
PRINT
100 ARCHIVO 1 LINEA 1
150 ARCHIVO 1 LINEA 2
500 ARCHIVO2 LINEA1
600 ARCHIVO2 LINEA2
700 ARCHIVO2 LINEA3
800 ARCHIVO2 LINEA4
900 ARCHIVO2 LINEA5
1000 ARCHIVO 3
1100 ARCHIVO 3
1200 ARCHIVO 3
```


INSERT

COPIA LINEAS DE UN ARCHIVO, BIEN SEA DE BIBLIOTECA O DE TRABAJO,. DEJANDOLAS EN EL RANGO ESPECIFICADO DENTRO DEL ARCHIVO DE TRABAJO.

ENTRE LAS OPCIONES VALIDAS Y USUALES ESTAN:

1.- INSERT <IDENTIFICADOR> AT END

2.- IN <RANGO><IDENTIFICADOR> AT <BASE> + <INCREMENTO>

3.- IN <IDENTIFICADOR> AT <BASE> + <INCREMENTO>

PARA COPIAR LINEAS DE UN ARCHIVO DE BIBLIOTECA SE DEBE USAR LA OPCION <IDENTIFICADOR>; SI SE OMITE, SE ENTIENDE QUE SE TRATA DEL ARCHIVO DE TRABAJO.

EN CASO DE NO ESPECIFICAR EL <RANGO> SERA COPIADO TODO EL ARCHIVO.

ES POSIBLE INSERTAR LA COPIA AL INICIO, EN MEDIO O AL FINAL UTILIZANDO 'AT' <BASE>+<INCREMENTO> O 'END', SEGUN SE DESEE.

EJEMPLOS

```
LIST CAMBIO/3
100 ARCHIVO 3
200 ARCHIVO 3
300 ARCHIVO 3
#
INSERT CAMBIO/1 200-END AT 350+5:L
#UPDATING
#
100 ARCHIVO 3
200 ARCHIVO 3
300 ARCHIVO 3
350 ARCHIVO 1 LINEA 2
355 ARCHIVO 1 LINEA 3
IN CAMBIO/2 AT 10+5:L
#UPDATING
#
10 ARCHIVO2 LINEA1
15 ARCHIVO2 LINEA2
20 ARCHIVO2 LINEA3
25 ARCHIVO2 LINEA4
30 ARCHIVO2 LINEA5
100 ARCHIVO 3
200 ARCHIVO 3
300 ARCHIVO 3
350 ARCHIVO 1 LINEA 2
355 ARCHIVO 1 LINEA 3
```


RESEQ

TIENE LA FINALIDAD DE ASIGNAR NUEVO NUMERO DE SECUENCIA A LAS LINEAS DE UN ARCHIVO DE TRABAJO, ESTO NO IMPLICA QUE CAMBIE SU ORDEN.

ES POSIBLE CAMBIAR EL NUMERO DE SECUENCIA TOTAL O PARCIALMENTE ESPECIFICANDO <RANGO> <BASE>+<INCREMENTO>. AL NO INDICAR ESTO, ASUME COMO NUMERO BASE 100 CON INCREMENTOS DE 100 O EL ULTIMO INCREMENTO QUE HAYA SIDO CONSIDERADO ANTES.

EJEMPLOS

```
LIST
10 ARCHIVO2 LINEA1
15 ARCHIVO2 LINEA2
20 ARCHIVO2 LINEA3
25 ARCHIVO2 LINEA4
30 ARCHIVO2 LINEA5
100 ARCHIVO 3
200 ARCHIVO 3
300 ARCHIVO 3
350 ARCHIVO 1 LINEA 2
355 ARCHIVO 1 LINEA 3
#
```

```
RESEQ 5001100
#UPDATING
#
LIST
500 ARCHIVO2 LINEA1
600 ARCHIVO2 LINEA2
700 ARCHIVO2 LINEA3
800 ARCHIVO2 LINEA4
900 ARCHIVO2 LINEA5
1000 ARCHIVO 3
1100 ARCHIVO 3
1200 ARCHIVO 3
1300 ARCHIVO 1 LINEA 1
1400 ARCHIVO 1 LINEA 2
#
```

OTROS COMANDOS USUALES

LOS COMANDOS HASTA AQUI DESCRITOS, SE PRESENTAN DE UNA MANERA MAS O MENOS LOGICA. LOS RESTANTES SE PRESENTARAN EN ORDEN ALFABETICO, SIENDO DENOMINADOS LOS PRIMEROS COMANDOS DE USUARIO Y LOS SEGUNDOS COMANDOS DE CONTROL

ADD Y COPY

SON EQUIVALENTES A LAS INSTRUCCIONES DE WFL (WORK FLO LANGUAGE) SIRVEN PARA HACER TRANSFERENCIAS DE ARCHIVOS ENTR DISPOSITIVOS DE ALMACENAMIENTO (DISCOS, CINTAS MAGNETICAS ETC).

EJEMPLOS

```
FILES
(GC87) ON UNAM3
, TIO ! SEQDATA
#
```

```
COPY (SL82)FIBO AS SERNUMFIBO FROM PACK TO DISK
#RUNNING 9011
#SCHD 9013
#BOT 9013 (GC87)WFLCODE
#BOT 9016 LIBRARY/MAINTENANCE
#9016 (GC87)SERNUMFIBO COPIED
#EOT 9016 LIBRARY/MAINTENANCE
#EOT 9013 (GC87)WFLCODE
#
```

FILES

(GC87) ON UNAM3

* TIO ! SEQDATA

* GERNUMFIBO ! FORTRAN

*

CHANGE Y TITLE

SON SINONIMOS, SE UTILIZAN PARA CAMBIAR EL NOMBRE A UN ARCHIVO DE TRABAJO, DE BIBLIOTECA O A UN DIRECTORIO.

EJEMPLOS

```
LOAD LASMOVIDAS
#WORKFILE LASMOVIDAS: DATA, 8 RECORDS, SAVED
*
CHANGE TO MOVIBLE
*
WHAT
#WORKFILE MOVIBLE: SOURCE IS (ISCA)LASMOVIDAS ON
                             IIMAS: DATA, 8 RECORDS
*
```



```
WHAT
#WORKFILE MOVIMIENTOS
*
CHANGE MOVIMIENTOS TO LASMOVIDAS
* (ISCA)MOVIMIENTOS CHANGED TO (ISCA)LASMOVIDAS ON
                                          IIMAS
*


TITLE NUMFIB  TO FIB
* (SL82)NUMFIB  (OBJECT) CHANGED TO
                    (SL82)FIB  ON PACK
*
```

CHARGE

SIRVE PARA IMPRIMIRLE UN IDENTIFICADOR O <CODIGO DE CARGA> A LA SESION DE TRABAJO, ESTE DEBE SER UN STRING DE 17 CARACTERES MAXIMO.

EJEMPLOS

```
CHARGE ULTIMO
#SESSION 7468 ET=1:36:17.7 PT=10.9 IO=9.6
#CONTINUE SESSION 7468 14:53:10 07/19/82
?WRU
#B7800:126 CANDE 31.2801 YOU ARE USUACU01(53)
#SESSION = 7468 USER = SL82 CHARGE = ULTIMO
```


DISCARD

SIRVE PARA REMOVER ARCHIVOS DE RECOBRO PRODUCIDOS EN SESIONES PREVIAS.

DISCARD <NUMERO DE RECOBRO> :- CUANDO SE DESEA DESCARTAR O DESTRUIR UN ARCHIVO, SI SON DOS O MAS SE DEBEN INDICAR SUS NUMEROS SEPARADOS POR COMAS.

EJEMPLOS

```
FILES
(SL82) ON PACK
. PUC
. . WFL : JOB
. CANDE
. . RECV350 : RECOVERYFILE
. FINAL : SEODATA
#
DISCARD 350
#
```

FAMILY

SE USA PARA CAMBIAR DE FAMILIA EN UNA SESION DE TRABAJO.

PARA CADA CLAVE DE USUARIO SE TIENE ASOCIADA POR OMISION UNA FAMILIA, RESULTANDO A VECES INSUFICIENTE POR RAZONES DE ESPACIO, SIENDO ENTONCES CONVENIENTE TRANSLADARSE DE FAMILIA.

EJEMPLOS

```
FAMILY PACK = DISK OTHERWISE DISK
#FAMILY PACK = DISK OTHERWISE DISK
```

```
FA DISK = PACK OTHERWISE PACK
#FAMILY DISK = PACK OTHERWISE PACK
```

```
FAMILY
#FAMILY DISK = PACK OTHERWISE PACK
```


FILES

SIRVE PARA LISTAR EL O LOS NOMBRE(S) DEL (O LOS) ARCHIVO(S) DE UNA CLAVE (LA PROPIA U OTRA) O DEL SISTEMA.

ALGUNAS OPCIONES SON LAS SIGUIENTES:

1.- FILES <IDENTIFICADOR> :- AL USARSE ESTA OPCION SE MOSTRARA SI ESTA PRESENTE EL ARCHIVO INDICANDONOS ADEMAS SU TIPO.

2.- FILES <IDENTIFICADOR> ON <NOMBRE DE FAMILIA> :- ESTA OPCION DEBE USARSE CUANDO EL O LOS ARCHIVO(S) QUE SE DESEA(N) CONSULTAR(SE) NO SE ENCUENTRA(N) EN LA FAMILIA DONDE SE TRABAJA.

3.- FILES <NOMBRE DE DIRECTORIO>

4.- FILES <NOMBRE DE DIRECTORIO>: PROFUNDIDAD

CUANDO SE UTILIZA <NOMBRE DE DIRECTORIO> SE TIENEN NIVELES DE <PROFUNDIDAD> ; SI SOLO INTERESAN AQUELLOS QUE TIENEN UNA DETERMINADA PROFUNDIDAD, ENTONCES SE DEBE ESPECIFICAR ':<PROFUNDIDAD>'.

EJEMPLOS

```
FILES LASMOVIDAS
(ISCA) ON IIMAS
   LASMOVIDAS : DATA
```

```
FILES
(SL02) ON PACK
. HE : ALGOL
. LEO
. . PASCAL01 : ALGOL
. MAR
. . CANDE : SEQDATA
. PUC
. . WFL : JOB
. HECT : ALGOL
. PILA : FORTRAN
. CANDE : DATA
. CADENA : ALGOL
. EXPERI
. . MENTO : ALGOL
. OBJECT
. . HE : JOVIALCODE
. . PILA : FORTRAN CODE
. . FUNCION : JOVIALCODE
. FUNCION : ALGOL
```


FIND

SE EMPLEA PARA LOCALIZAR UN TEXTO, HACIENDO LA BUSQUEDA DE UNA CUERDA DE CARACTERES EN UN ARCHIVO DE TRABAJO O EN UNO DE BIBLIOTECA.

LOS <DELIMITADORES> PUEDEN SER CUALQUIER CARACTER ESPECIAL, Y SIRVEN PARA MARCAR EL PRINCIPIO Y FINAL DEL TEXTO, ESTOS DEBEN SER HOMOGENEOS.

OPCIONES PERMITIDAS:

1.- FIND LITERAL <DELIM><TEXTO><DELIM>

2.- FIND<DELIM><TEXTO><DELIM><RANGO>

3.- FIND <CANTIDAD><DELIM><TEXTO><DELIM>

EL TEXTO DEBE SER UN STRING EN EL QUE LOS BLANCOS INNECESARIOS NO SON SIGNIFICATIVOS A MENOS QUE SE USE LA OPCION 'LITERAL'. LA OPCION <CANTIDAD> SIRVE PARA LIMITAR EL NUMERO TOTAL DE OCURRENCIAS. EL <RANGO> LIMITA LA BUSQUEDA A LAS LINEAS COMPRENDIDAS EN EL.

4.- FIND <DELIM><TEXTO><DELIM><IDENTIFICADOR>

5.- FIND<DELIM><TEXTO><DELIM><IDENTIFICADOR> ON <NOMBRE DE FAMILIA>

6.- FIND<DELIM><TEXTO><DELIM>:FILE <IDENTIFICADOR>

SI SE ESPECIFICA <IDENTIFICADOR> LA BUSQUEDA SE EFECTUARA EN UN ARCHIVO DE BIBLIOTECA. SI ESTE NO SE ENCUENTRA EN LA FAMILIA EN QUE SE ESTA TRABAJANDO DEBE MENCIONAR 'ON <NOMBRE DE FAMILIA>'.

ES POSIBLE ALMACENAR LAS LINEAS EN LAS QUE SE ENCONTRO EL <TEXTO> DENTRO DE UN NUEVO, ARCHIVO USANDO LA OPCION ':FILE <IDENTIFICADOR DESTINO>'.

7.- FIND<DELIM><TEXTO><DELIM>:TEXT

LA OPCION 'TEXT' DESPLIEGA LA INFORMACION DE LAS LINEAS EN LAS QUE EL <TEXTO> FUE ENCONTRADO; SI NO SE USA ESTA, SOLO APARECERAN LOS NUMEROS DE SECUENCIA DE DICHAS LINEAS.

EJEMPLOS

```
FIND/ESTA/:T
#WORKFILE MOVIMIENTOS
400  ESTA ES LA PRIMERA LINEA
500  ESTA ES LA SEGUNDA LINEA
600  ESTE ARCHIVO ESTA HECHO CON EL OBJETO DE
#
```

```
FIND/ESTA/
#WORKFILE MOVIMIENTOS
400,500,600
```



```
FIND/#/HECANDE:T
#FILE (SL02)HECANDE ON PACK
25300 SA AS #<IDENTIFICADOR DE ARCHIVO>
#
```

```
FIND,(*),CANDE1:T
#FILE (SL02)CANDE1 ON PACK
40300 O (*), QUE PROVOCA QUE LA LISTA
43250 SI SE ESPECIFICA (*), DE LO CON-
43800 O CONSIDERANDO (*) PARA
#
```

```
FIND/2.1 IO=1.5/:T
#WORKFILE CARCANDE
28200 #ET=12.1 PT=2.1 IO=1.5
#
```

LFILES

LISTA ATRIBUTOS DE LOS ARCHIVOS DE BIBLIOTECA DEL USUARIO O DE OTROS USUARIOS, DEPENDIENDO DE LA SEGURIDAD QUE ELLOS TENGAN.

SE PUEDEN EMPLEAR :

1.- LFILES <IDENTIFICADOR>

2.- LFILES <NOMBRE DE DIRECTORIO>

CON LA PRIMERA OPCION <IDENTIFICADOR> SOLO SE LISTAN LOS ATRIBUTOS DE DICHO ARCHIVO; DE FORMA SEMEJANTE, LA OPCION <NOMBRE DE DIRECTORIO> LISTARA LOS DE ESE DIRECTORIO, EN CASO DE NO ESPECIFICAR NINGUNA DE ESTAS, SERAN MOSTRADOS LOS ATRIBUTOS DE TODOS LOS ARCHIVOS DE BIBLIOTECA DEL USUARIO.

3.- LFILES:PRINTER :- ESTA OPCION NOS DA UN LISTADO POR IMPRESORA, EN LUGAR DE SALIR POR TERMINAL.

4.- LFILES:ABBREVIATED :- MUESTRA EL LISTADO DE ATRIBUTOS EN FORMA ABREVIADA, NO TAN DETALLADA.

EJEMPLOS

```
LFILES:ABBREVIATED
#SCHD 1745
#RUNNING 1745
(SL82) : DIRECTORY  ON PACK
.  PUC : DIRECTORY
.  .  WFL :JOBSYMBOL
.  FINAL : SEQDATA
.  HECTOR : ALGOLSYMBOL
.  OBJECT : DIRECTORY
.  .  HECTOR : JOVIALCODE
.  .  FIBONACCI : FORTRANCODE
.  EJEMCANDE : SEQDATA
.  MANUALWFL : SEQDATA
.  FIBONACCI : FORTRAN
```

NEWS

ESTE COMANDO SIRVE PARA LISTAR EL ESTADO ACTUAL DE UN ARCHIVO DE "NOTICIAS" BAJO EL CONTROL DEL SISTEMA OPERATIVO.

EJEMPLOS

```
NEWS
##  REMOCION DE ARCHIVOS NO ACCESADOS TECLEE NEWS ##
DEBIDO A LA DEMANDA DE ESPACIO EN PACK'S
A PARTIR DEL 28 DE JUNIO SE REMOVERAN TO-
DOS LOS ARCHIVOS DE LOS PACK'S LOS CUALES
NO HAN TENIDO ACCESO DURANTE LOS ULTIMOS -
15 (QUINCE) DIAS, ESTE PROCEDIMIENTO SE EFEC-
TUARA CADA VEZ QUE EXISTA FALTA DE ESPACIO -
EN CUALQUIERA DE LAS FAMILIAS EXISTENTES EN
EL SISTEMA.

C.S.C.                    23/JUNIO/82
```


RECOVER

PERMITE CARGAR UN ARCHIVO DE RECOBRO COMO ARCHIVO DE TRABAJO. ESTOS SE GENERAN POR INTERRUPCION DE LA SESION PREVIA O POR LA ACCION DEL COMANDO 'SAVE RECOVERY'.

PARA NO SER RECHAZADO EL COMANDO 'RECOVER' ES NECESARIO NO TENER PRESENTE UN ARCHIVO DE TRABAJO. EL NUMERO DE RECOBRO ESPECIFICA AQUEL ARCHIVO QUE SE QUIERE RECUPERAR.

EJEMPLOS

```
HELLO SL82/XX
#DEFAULT PRINT DESTINATION=SITE
#RECOVERY DATA ON PACK
  570 HECANDE (06/11/82)
#SESSION 9018 09:25:57 06/12/82
REC 570
#WORKFILE HECANDE: SEQ
#
WHAT
#WORKFILE HECANDE: SEQ, 245 RECORDS (THRU 25500)
```

SECURITY

SIRVE PARA CAMBIAR LOS ATRIBUTOS DE SEGURIDAD DE UN ARCHIVO DE BIBLIOTECA.

SE PUEDEN EMPLEAR LAS OPCIONES:

1.- SEC OBJECT <IDENTIFICADOR><SEGURIDAD>

2.- SEC SOURCE <IDENTIFICADOR><SEGURIDAD>

CON LA OPCION 'OBJECT' SE CAMBIA LA SEGURIDAD DEL ARCHIVO OBJETO, CON LA OPCION 'SOURCE' SE CAMBIA LA DEL ARCHIVO FUENTE; SI NO SE ESPECIFICA NINGUNA DE ESTAS OPCIONES, AMBAS VERSIONES (FUENTE Y OBJETO) SERA CAMBIADA LA SEGURIDAD (SIEMPRE QUE SEAN DEL TIPO LENGUAJE DE PROGRAMACION) , POR OMISION, LA SEGURIDAD BAJO LA CUAL QUEDARA GUARDADO SERA 'PRIVATE I/O'.

F3

EJEMPLOS

```
SEC FIBONACCI CLASSA
*  (JA94)FIBONACCI ON UNAM4 SECURITY CHANGED

SEC FIBONACCI PUBLIC
*  (SL82)FIBONACCI ON PACK (% OBJECT) SECURITY CHANGE
```


SPLIT

SIRVE PARA DAR POR TERMINADA UNA SESION Y DAR EL INICIO A OTRA, SIN DESCONECTARSE DEL SISTEMA.

SI DENTRO DE UNA SESION SE MANDA EJECUTAR UN PROGRAMA Y DESPUES IMPRIMIR, SALDRA EN UNA SOLA IMPRESION (UN SOLO LISTADO) PERO SI DESPUES DE EJECUTARSE EL PROGRAMA Y ANTES DE MANDARLO IMPRIMIR SE USA EL COMANDO 'SPLIT', SE TENDRAN DOS LISTADOS (UNO DE LA EJECUCION Y OTRO DE RESULTADOS).

EJEMPLOS

```
SPLIT
*SPLIT SESSION 4879 ET=49:50.2 PT=13.1 IO=6.9
*NEW SESSION 5529 14:21:47 06/15/82

SPLIT
*SPLIT SESSION 7468 ET=11:04.1 PT=0.4 IO=0.3
*NEW SESSION 8722 15:04:15 07/19/82
```

START

EJECUTA UN TRABAJO (JOB) DE WFL EN MODO 'BATCH' DE PROCESAMIENTO.

START <IDENTIFICADOR> !- INDICA UN ARCHIVO QUE CONTIENE INSTRUCCIONES EN WFL ,SIENDO ESTE UN ARCHIVO DE BIBLIOTECA ,EL <IDENTIFICADOR> NO ES NECESARIO SI SE TRATA DEL ARCHIVO DE TRABAJO.

ES POSIBLE CAMBIAR EL CARACTER ILEGAL ('?') POR OTRO, PERO DEBE SER ESPECIFICADO CON EL COMANDO DE CONTROL'?=<CARACTER>'.

EJEMPLOS

```
MAKE TAREA JOB
#WORKFILE TAREA1 JOB
SEQ
100 RUN $SERVICIO/TIA;FILE FUENTE(TITLE=FINAL)
200 END
300
$
```



```
SAVE
#UPDATING
#WORKSOURCE TAREA SAVED
START TAREA
#RUNNING 1563
```

```
START TAREA
#RUNNING 1573
#JOB 1574 IN D 04
$
#1574 BOJ  'RUN $SERVICIO"
#1574 1578 BOT $SERVICIO/TIA
#1578 OLD FORMAT ROUTINES TO BE DEIMPLEMENTED ON 3.2
#1574 1578 EOT SERVICIO/TIA
#1574 1574 EOJ JOB RUN $SERVICIO
$
```

TYPE

SE USA PARA CAMBIAR LA ESTRUCTURA INTERNA DE LAS CARACTERISTICAS DE UN ARCHIVO.

TYPE <IDENTIFICADOR> TO <TIPO> 1- SI SE TRATA DE UN ARCHIVO DE BIBLIOTECA SE DEBE ESPECIFICAR <IDENTIFICADOR>, DE LO CONTRARIO SE ASUME EL DE TRABAJO.

LA PALABRA 'TO' ES OPCIONAL, NO ASI <TIPO>.

EL HECHO DE CAMBIAR EL TIPO DEL ARCHIVO NO IMPLICA QUE SU CONTENIDO CAMBIE.

EJEMPLOS

```
TYPE FIBONACCI TO PL/1
$UPDATING
$
```

```
TY FIBONACCI TO FORTRAN
$
```


UPDATE

ACTUALIZA UN ARCHIVO DE TRABAJO, DANDO DE ALTA TODAS LAS MODIFICACIONES QUE HAYAN SIDO HECHAS, TALES COMO: CAMBIOS, INSERSIONES, REEMPLAZOS, SUPRESIONES DE LINEA, ETC.

ALGUNOS COMANDOS IMPLICAN UNA FORZOSA ACTUALIZACION, ESTOS SON: RUN, COMPILE, EXCLUDE, MERGE, INSERT, MOVE, RESEQ, RANGE, EXECUTE, BIND, TYPE, FIND, REPLACE Y SAVE.

EJEMPLOS

```
UPDATE
$
```

```
MOVE 94307-94309 TO 9504+2
$UPDATING
$UPDATING
$COMPILING
$ET=6.8 PT=1.1 IO=1.3
.RUNNING 2470
$?
"HOLA QUE TAL"
$ET=12.9 PT=0.3 IO=1.0
```

WHAT

SIRVE PARA PROPORCIONAR INFORMACION REFERENTE AL ARCHIVO DE TRABAJO Y EL ESTADO QUE GUARDA, OCASIONANDO MENSAJES TALES COMO:

NOMBRE Y TIPO DEL ARCHIVO ,
NUMERO DE LINEAS,
NUMERO DE SECUENCIA DE LA ULTIMA LINEA,
ARCHIVO FUENTE SALVADO, ETC.

EJEMPLOS

```
WHAT
#WORKFILE HECANDE1 SEQ, 245 RECORDS (THRU 25500)
```

```
WHAT
#WORKFILE MOVIBLE1 SOURCE IS (ISCA)LASMOVIDAS ON I1
LIST
100  OBSERVA LOS MOVIMIENTOS QUE SOBRE
200  TODO EL ARCHIVO SE REALIZARAN
300  LISTO, EMPEZAREMOS.
400  MIRA ESTA ERA LA PRIMERA LINEA
500  MIRA ESTA ERA LA SEGUNDA LINEA
600  ESTE ARCHIVO MIRA ESTA HECHO CON EL OBJETO DE
```



```
700  REALIZAR MOVIMIENTOS CON SUS LINEAS
800  VEAMOS
```


WRITE

MANDA UN LISTADO DE UN ARCHIVO, BIEN SEA DE TRABAJO O DE BIBLIOTECA, HACIA IMPRESORA.

ALGUNAS DE LAS OPCIONES VALIDAS SON:

1.- WRITE:NUMBERED

2.- WRITE:UNSEQUENCED

3.- WRITE:DOUBLE

4.- WRITE:SQUASHED

ES POSIBLE IMPRIMIR TOTAL O PARCIALMENTE UN ARCHIVO, CON SECUENCIA O SIN ELLA (NUMBERED O UNSEQUENCED, RESPECTIVAMENTE) A DOBLE ESPACIO (DOUBLE) O SUPRIMIENDO BLANCOS (SQUASHED).

EJEMPLOS

```
W FIBONACCI:D
#RUNNING 8861
#
```

```
WRITE
#RUNNING 9905
#
```

```
WRITE DIREC/=:D
#RUNNING 9996
#
```


COMANDOS DE CONTROL

ESTOS VAN PRECEDIDOS DE UN CARACTER ILEGAL, PARA DISTINGUIRLOS DE LOS DE USUARIO, SIRVEN, COMO SU NOMBRE LO INDICA, PARA CONTROLAR EVENTOS, CUANDO OCURREN ERRORES O CONDICIONES INESPERADAS SOBRE ELLOS.

ESTOS TIENEN LA FINALIDAD DE OTORGARLE AL USUARIO PRIORIDAD PARA GANAR EL CONTROL SOBRE EL PROCESO DE PROGRAMAS O ESTADOS SOBRE LA RED DE COMUNICACION DE DATOS O SOBRE DISPONIBILIDAD DE ALMACENAMIENTO PRINCIPAL.

MUESTRA AL USUARIO LA LISTA DE TAREAS TERMINADAS MAS RECIENTEMENTE BAJO SU CLAVE.

SON VALIDAS LAS SIGUIENTES OPCIONES:

1.- (?C) <NUMERO LOGICO DE ESTACION>

2.- (?C) *

ESTAS OPCIONES, MUESTRAN LA LISTA DE TAREAS DE UNA ESTACION EN PARTICULAR, SI SE OMITEN <NUMERO LOGICO DE ESTACION> O EL '*', LA LISTA SE VOLVERA MAYOR.

EJEMPLOS

RUN

```
#COMPILING 2091
#ET=17.6 PT=1.2 IO=1.4
#SCHD 2099
 ?C
---- COMPLETED ENTRIES ----
 #2000/2091 EOT  FORTRAN (SL82)CANDE/CODE350 C.I PACK
#RUNNING 2099
#ET=32.3 PT=0.4 IO=0.9
 ?C
---- COMPLETED ENTRIES ----
 #2000/2099 EOT (SL82)CANDE/CODE350 ON PACK
  2000/2091 EOT FORTRAN (SL82)CANDE/CODE350 ON PACK
```

(?CHARACTER)

CAMBIA LOS CARACTERES DE CONTROL: DE FIN-DE-LINEA, SUPRESION DE LINEA, RETROCESO, SI SE USA SIN PARAMETROS, DESPLIEGA EL CONJUNTO DE CARACTERISTICAS QUE REALIZAN LAS FUNCIONES DESCRITAS.

EJEMPLOS

```
?CHARACTER
#BACK = BS, CONTROL = ?, DEL = DEL, END = CR
```

(?COUNTS)

PERMITE CONOCER CUAL ES EL TOTAL DE ESTACIONES ACTIVAS Y CONECTADAS A TRAVES DE CANDE.

EJEMPLOS

```
?COUNTS
#5 TASKS,2 WORKERS/22 STATIONS ACTIVE,64 ATTACHED
```

```
?COUNTS
#5 TASKS,0 WORKERS/20 STATIONS ACTIVE,64 ATTACHED
```


(?CS)

ESTE COMANDO SE UTILIZA PARA CONOCER EL ESTADO DE UNA COMPILACION DE UN PROGRAMA, SI LA COMPILACION SE ACTIVO DESDE OTRA ESTACION, SE DEBE INDICAR EL <NUMERO DE MEZCLA>.

EJEMPLOS

```
?CS
SEQ = 00000018, NO ERRORES
#ET=12.1 PT=2.1 IO=1.3
```

(?CU)

SIRVE PARA AVERIGUAR EL USO TOTAL Y ACTUAL DE MEMORIA PARA UNA TAREA, ESTA PUEDE SER EJECUTADA EN LA PROPIA TERMINAL O EN OTRA, EN ESTE ULTIMO CASO DEBE AGREGARSE EL <NUMERO DE MEZCLA>.

EJEMPLOS

```
C
#COMPILING 2265
?CU
2265 CORE:   TOTAL    SAVE
  STACK       8348    4334
  CODE       10436     567
    TOTAL    10784    4091
#ET=16.4 PT=1.3 IO=1.3
```


?DENY Y ?END).

NHABILITAN LOS REQUERIMENTOS DE ENTRADA DE DATOS, ENVIANDO NA SENAL DE FIN-DE-ARCHIVO A LOS PROGRAMAS QUE TENGAN UN AR-HIVO DE ENTRADA ABIERTO.

JEMPLOS

```
R
#RUNNING 2430
#?
?END
#243A EOF NO LABEL:LEC @ 2771002213*
#I-DS @ 44222400, 44835200, 44832400, 00000350
#ET=31.0 PT=0.3 IO=1.4
```

(?DS).

DA POR TERMINADO UN TRABAJO O UNA TAREA, EL(LA) CUAL PUDO HABER SIDO ARRANCADO(A) DESDE LA TERMINAL O DESDE OTRA ESTACION, TENIENDO QUE ESPECIFICAR EN ESTE SEGUNDO CASO EL <NUMERO DE MEZCLA>.

EJEMPLOS

```
R
RUNNING 2422
?WHY
STATUS OF TASK 2422 AT 15:01:25
PRIORITY = 50
ORIGINATION: LSN 35
STACK STATE: WAITING ON AN EVENT
PROGRAM NAME: (SLB2)CANDE/CODE350 ON PACK
?DS
#2422 OPERATOR DSED @ 26A:020:1
#0-DS @ 44835200, 00000350, 44908400, 00000350
#ET=54.7 PT=0.4 IO=1.0
```


(?JA).

SIRVE PARA OBTENER UN LISTADO DE LOS TRABAJOS QUE SE ESTAN EFECTUANDO BAJO ALGUNA CLAVE DE USUARIO, QUE PUEDE SER LA PROPIA O ALGUNA OTRA EN ESPECIAL, LOS TRABAJOS PUEDEN ESTAR ACTIVOS EN LA TERMINAL , EN ALGUNA OTRA, O EN DISTINTAS TERMINALES SIMULTANEAMENTE.

ALGUNAS OPCIONES :

1.- (?JA) <NUMERO LOGICO DE ESTACION>

2.- (?JA) *

SI NO SE ESPECIFICA <NUMERO LOGICO DE ESTACION> O (*), SE MOSTRARA EL TOTAL DE TRABAJOS ACTIVOS.

EJEMPLOS

```
START TAREA
RUNNING 4495
#JOB 4496 IN Q 04
#
#4496 BOJ "COMPILE FIBONACCI FORTRAN"
#4496/4497 BOT FORTRAN FIBONACCI
TJA
---- JOB STRUCTURE ----
#4496 JOB 45 COMPILE FIBONACCI FORTRAN
#4..4497 45 FORTRAN ON PACK (CANDE)FIBONACCI
```


(TSQ)

LISTA LOS TRABAJOS QUE ESTAN EN ESPERA DE EJECUCION EN CADA COLA, SOLO SERA MOSTRADO UN TRABAJO, A MENOS QUE SE INDIQUE UNA COLA EN PARTICULAR.

SE PUEDE LIMITAR LA LISTA CONSIDERANDO UN <NUMERO LOGICO DE ESTACION> O UN (*) PARA LA TERMINAL PROPIA (LOCAL).

EJEMPLOS

```
TSQ
QUEUE 1:
 NO ENTRIES
QUEUE 2:
 NO ENTRIES
QUEUE 3:
 NO ENTRIES
QUEUE 4:
 3789 52 RUN FIBONACCI (#0036)
QUEUE 5:
 NO ENTRIES
QUEUE 66:
 NO ENTRIES
#
```

```
?SQ 66
QUEUE 66:
 NO ENTRIES
#
```

```
?SQ 4
QUEUE 41
 3789 52 RUN FIBONACCI (#0009)
 3817 52 RUN FIBONACCI (#0010)
```


(?SS Y ?TO)

SIRVEN PARA ENVIAR MENSAJES DE UNA A OTRA TERMINAL O AL OPERADOR DEL SISTEMA, ASI COMO AL USUARIO DE UNA CLAVE.

PARA UNA TERMINAL EN PARTICULAR SE ESPECIFICA SU <NUMERO LOGICO DE ESTACION> O SU <IDENTIFICADOR DE ESTACION>, PARA EL OPERADOR SE INDICA 'SPO'.

PARA ENVIAR MENSAJES A UN USUARIO EN PARTICULAR SE DEBE USAR '?TO <CLAVE DE USUARIO>', RECIBIENDO EL MENSAJE TODOS Y CADA UNO DE LOS USUARIOS QUE ESTAN CONECTADOS AL SISTEMA BAJO ESA CLAVE.

EJEMPLOS

```
?SS 34 QUIEN ESTA POR AHI, SOY A.C.L
#
#14:28 FROM VR81 ON 34: DISCULPA QUIEN ES ACL ???
#
```

```
?TO VR81 SOY AGUSTIN Y TU, ESTE ES UN MENSAJE OK
#
#14:30 FROM VR81 ON 34: OK.
#
```

(?STATUS)

PERMITE CONOCER EL ESTADO QUE GUARDA UNA TERMINAL DENTRO DEL SISTEMA, CON RESPECTO A TAREAS, TRABAJOS O COMANDOS CANDE.

SI NO SE USA NINGUNA OPCION, SE ASUME QUE SE DESEA SABER EL ESTADO DE LA TERMINAL Y DE LA TAREA O TRABAJO QUE SE ESTE EJECUTANDO. EN CASO CONTRARIO SE DEBE EMPLEAR EL <NUMERO LOGICO DE ESTACION>.

EJEMPLOS

```
?STA
#14151 7TH IN TASK QUEUE
```

```
?STA
#14146 9TH IN TASK QUEUE
```

```
?STA
#14142 3RD IN TASK QUEUE
```



```
?STATUS 35
FCIENCMAT3(35)=0133:0 RDY ENAB ATT
```

```
?STA 53
USUACU01(53)=1:0:0 RDY ENAB ATT
```

(?TI)

MUESTRA LOS TIEMPOS DE PROCESAMIENTO, DE ENTRADA/SALIDA Y DE ESPERA QUE ESTAN SIENDO UTILIZADOS POR UNA TAREA.

EJEMPLOS

```
COMPILE
#COMPILING 6149
?TI
TIMES FOR 6149
  PROCESS=00:00:00
  IO     =00:00:00
  ELAPSED=00:00:04
#ET=5.2 PT=0.5 IO=0.6
```


(?TIME)

SIRVE PARA OBTENER COMO RESPUESTA LA HORA, EL DIA Y LA FECHA.

EJEMPLOS

```
?TIME
#3:07 PM THURSDAY, JULY 15, 1982
#

?TIME
#2:45 PM MONDAY, JULY 19, 1982
```

(TWD).

PROPORCIONA, AL USUARIO, EL DIA Y LA FECHA.

EJEMPLOS

```
?WD
DATE IS THURSDAY JUL 15,1982 (82196) 15:06:48


?WD
DATE IS MONDAY JUL 19,1982 (82200) 14:48:33
```


(?WHERE).

ES UTIL PARA INVESTIGAR EN DONDE SE ENCUENTRA ALGUN USUARIO.

EJEMPLOS

```
?WHERE MCB1
# MCB1 ON DIMEFI14(92)
# MCB1 ON DIMEFI13(91)
# MCB1 ON USUACU04(35)


?WH AJB0
# AJB0 ON CECAFI1(34)
```

(?WHY O ?Y).

SON SINONIMOS Y SU OBJETIVO ES DESPLEGAR EL ESTADO QUE GUARDA LA EJECUCION DE UNA TAREA, ESTA PUEDE SER REALIZADA DESDE LA TERMINAL O DESDE OTRA .

EJEMPLOS

```
R
RUNNING 2422
?WHY
STATUS OF TASK 2422 AT 15:01:25
PRIORITY = 50
ORIGINATION: LSN 35
STACK STATE: WAITING ON AN EVENT
PROGRAM NAME: (SL82)CANDE/CODE350 ON PACK
```



```
?Y
STATUS OF TASK 6182 AT 07:54:42
PRIORITY = 50
ORIGINATION: LSN 35
STACK STATE: WAITING ON AN EVENT
COMPILER NAME: SYSTEM/FORTRAN ON PACK
PROGRAM NAME: (CANDE)OBJECT/FORTRANFILE
RSVP: OPERATOR STOPPED
REPLY: OK,DS
?DS
#6182 GOING
#6182 OPERATOR DSED @ 003:00C0:04
#O-DS @ 99656300.
#ET=27.6 PT=0.8 IO=0.6
```

(?WRU).

SIRVE PARA IDENTIFICAR EL NOMBRE Y NUMERO LOGICO DE UNA TERMINAL Y CONOCER EL USUARIO QUE SE ENCUENTRA EN SESION.

EJEMPLOS

```
?WRU
#87800:126 CANDE 31.280! YOU ARE FCINCHAT3(35)
#SESSION = 2254 USER = SL82
```

```
?WRU
#87800:126 CANDE 31.280! YOU ARE USUACU01(53)
#SESSION = 7468 USER = SL82 CHARGE = AGUSTIN
```


(?WT)

SU FUNCION ES LA DE MOSTRAR LA FECHA Y HORA.

EJEMPLOS

```
?WT
DATE IS THURSDAY JUL 15,1982 (82196) 15:13:26
```

```
?WT
DATE IS MONDAY JUL 19,1982 (82200) 14:51:00
```

C A N D E

- INTRODUCCION
- ¿QUE ES CANDE?
- VENTAJAS, DESVENTAJAS
- TERMINOLOGIA
- OBSERVACIONES
- PREPARACION PARA UTILIZAR EL SISTEMA
- DIAGRAMA DE FLUJO
- COMANDOS

MAYO DE 1983

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

DR. OCTAVIO RIVERO SERRANO
Rector

LIC. RAUL BEJAR NAVARRO
Secretario General

C.P. RODOLFO COETO MOTA
Secretario General Administrativo

DR. JORGE HERNANDEZ Y HERNANDEZ
Secretario de Rectoría

LIC. IGNACIO CARRILLO PRIETO
Abogado General

ING. JORGE GIL MENDIETA
Director General del Programa
Universitario de Cómputo

RESPONSABLE DE LA PUBLICACION
MA. GUADALUPE CASTELLANOS VAZQUEZ GIL

PROGRAMA UNIVERSITARIO DE COMPUTO
FORMACION DE RECURSOS HUMANOS
1982

I N D I C E

SECCIONES

SECCION I

I N T R O D U C C I O N

La tecnología moderna del área de computación, entre otros elementos, permite el acceso a equipos de cómputo a través de terminales remotas, dispositivos que enlazan, desde un punto relativamente lejano a un equipo central, al usuario con la computadora en cuestión, para que éste trabaje con los recursos y elementos que provee dicho equipo. Esta modalidad de acceso - por sus facilidades y ventajas, se ha difundido rápidamente en la mayoría de los equipos de cómputo que existen actualmente - en el mercado.

Para utilizar las terminales conectadas a un equipo Central, - es necesario que la persona interesada conozca el EDITOR.

En la Universidad Nacional Autónoma de México, se cuenta con - equipo Burroughs B-7800, mismo que cuenta con un editor llamado CANDE.

El presente trabajo constitutye una recopilación de los comandos más usuales para manejar CANDE. Fue realizado para los cursos que se imparten en el Programa de Formación de Recursos Humanos, del Programa Universitario de Cómputo, tomando como punto de partida las notas que los profesores de dichos cursos desarrollaron y conjuntamente con un grupo de becarios de programa de formación de recursos humanos.

La forma y orden en que están tratados los temas en el presente texto, consideramos que es sumamente útil a personas que por - primera vez harán uso de CANDE, por lo que este material pudiera utilizarse en cursos básicos sobre este editor.

Este texto fué realizado con la colaboración de los siguientes becarios del Programa Universitario de Cómputo (P.U.C.):

- Besprosvany Fridzon Jaime
- De la Torre Beltrán Nelly
- Figueroa Servín Olga
- García Flores Marco Antonio
- Granados Cisneros María Rebeca
- Mendoza Romero David
- Nolasco Orozco Esmeralda
- Pérez Cabrera Rafael
- Ramírez Márquez Eduardo
- Reyes Navarrete Armando Alejandro
- Villarreal Cueva Alberto
- You Jianyu

SECCION II

II) QUE ES CANDE?

CANDE significa Command and Edit, es un editor para las series Burroughs B7000/B6000 que provee capacidades de preparacion y actualizacion de archivos en un medio interactivo orientado a terminal es decir, permite al usuario tener acceso al sistema de tiempo compartido, actuando en forma interactiva desde una terminal de teleproceso con la computadora.

Los archivos creados a traves de Cande pueden ser editados, protegidos y utilizados posteriormente por el usuario, por un programa o por otros usuarios, pudiendo ser transferidos de un dispositivo a otro tales como disco fijo, disco removible y cinta magnetica.

Ademas, los programas podran ser manipulados, compilados y ejecutados en el momento deseado, siempre y cuando hayan sido almacenados previamente.

SECCION III

III) VENTAJAS Y DESVENTAJAS.

A) Ventajas.

Cande es un editor muy util que permite preparar archivos en forma flexible y eficiente ademas de ahorrar tiempo y brindar facilidad de edicion y de apreciacion inmediata de resultados. Asi como tambien permite mandar ordenes a la maquina para que esta realice las acciones requeridas.

Lo anterior se debe a que Cande trabaja en un medio interactivo, es decir, un medio en el cúal el usuario interactua con la computadora en forma inmediata (a manera de dialogo) y como tal recibe respuesta inmediata (olvidandose para esta consideracion de los tiempos de espera por las colas), apreciando de esta forma el resultado de sus acciones para que, de acuerdo a estas, tome decisiones sobre acciones futuras.

Por lo tanto, se puede decir que las ventajas principales son :

i) Ahorro de tiempo.

ii) Facilidad, eficiencia y flexibilidad en la preparacion de archivos.

iii) Resultados inmediatos.

iv) T de deci i inmediata.

B) Desventajas.

Las desventajas que presenta el earleo de Cande se pueden resumir en las siguientes :

i) Se requiere una terminal (generalmente se encuentran ocupadas).

ii) Se requiere poseer una clave de cierta prioridad.

iii) Se requiere conocer Cande y estar familiarizado con el.

SECCION IV

IV) TERMINOLOGIA.

A) Clave del usuario.

Esta clave es una identificacion del usuario, indispensable para poder hacer uso de los servicios de computo del sistema Burroughs.

El sistema requiere de esta para que el usuario pueda entrar en sesion y para que la maquina sepa quien es el usuario. Por lo tanto el sistema checa internamente su validez asi como el tipo de clave de que se trate (estudiante, investigador, etc.).

La clave se compone de dos partes :

USERCODE.- Esta consta de dos letras (inicial del nombre y el apellido) y dos numeros (aleatorios) y el usuario la utiliza para identificarse externamente (para recoger listados, comunicacion entre usuarios, reservar tiempo para uso de terminal, etc.).

Ademas es utilizado internamente para identificar archivos en la libreria del usuario (ver definicion de directorio).

PASSWORD.- Esta es una contrasena secreta utilizada por el usuario conjuntamente con el usercode para identificarse con la maquina y solo debe ser conocida por el propietario de la clave. Es decir, es la parte secreta de la clave del usuario.

B) Sesion.

Una sesion de teleproceso es el periodo que transcurre entre la identificacion del usuario con la maquina hasta el momento en que se desde de ella o solicita una nueva sesion. (Ver seccion de preparacion para hacer uso de Cande, comando BYE y comando SPLIT).

Inicia sesion :

```
        HELLO
        #6700:1268 CANDE 31.280; YOU ARE CIFCA(03)
        #ENTER USERCODE PLEASE
        ARSO
        #ENTER PASSWORD PLEASE.
        APACELI
        #DEFAULT PRINT DESTINATION=SITE
        SEMINARIOS PARA LOS MESES DE MAYO Y JUNIO
[1]---> #SESSION 5248   11:12:26   85/12/02
```

Termina sesion :

```
        BYE
[2]---> #END SESSION 5248  ET=11.1 PT=0.8 IO=0.6
        #USER = ARSO   11:12:38   85/12/02
```

[1] : Da el numero de la sesion asignada al usuario. Hasta este momento el usuario entra a sesion.

[2] : Indica el fin de la sesion con el numero que le asigno (en este caso 5248). Este paso es necesario antes de apagar la maquina.

C) Archivo de trabajo (WORKFILE).

Es un archivo que acaba de ser creado (con el comando MAKE) o una copia de un archivo ya almacenado anteriormente (con el comando SAVE) en la biblioteca del usuario, pero que se ha traido presente de disco removible (con el comando GET).

Es el archivo con el cual se va a trabajar (editar, correr, etc.).

D) Archivos de biblioteca.

Cuando un archivo de trabajo es guardado (con el comando SAVE) pasa a ser un archivo de biblioteca, es decir, sera almacenado en la biblioteca identificada con el usercode del usuario (biblioteca del usuario) hasta que sea removido (con el comando REMOVE).

No se puede alterar el contenido de un archivo de biblioteca a traves de CANDE hasta que el archivo este presente, es decir, hasta que sea un archivo de trabajo (con el comando GET).

En este caso la version de biblioteca no es afectada por los cambios hechos en el archivo de trabajo hasta que este sea guardado nuevamente (con el comando SAVE), con lo cual la nueva version reemplazara a la version anterior que reside en la biblioteca.

E) Directorio.

Un directorio consiste de un conjunto de archivos de biblioteca. Cada clave en particular tiene su propio directorio, el cual integra la biblioteca del usuario.

Estructura de un directorio.

Considerese el siguiente fragmento de la biblioteca de archivos de un usuario :

```
        (RM81) ON PACK
          . PASCAL
          . .  PROG : ALGOL
          . .  PROG7 : ALGOL
          . .  TAREA1 : ALGOL
          . FORTRAN1 : SEQDATA
[1]---)   . FORTRAN2 : SEQDATA
```


Este fragmento nos indica que existen los siguientes archivos :

PASCAL/PROG

PASCAL/PROG7

PASCAL/TAREA1

FORTRAN1

FORTRAN2 .

(estos nombres son, los que el usuario le dio a cada archivo el momento de crearlos con el comando MAKE).

[1] : De aqui se puede interpretar el significado del "." (punto). Es decir, este va directamente abajo de la palabra que reemplaza (le sustituye).

Se puede apreciar que bajo el USERCODE(RM81), todos los archivos llevan el ".", esto significa que la maquina le antepone automaticamente el USERCODE al nombre de todos y cada uno de los archivos. Se puede ver que existen tres archivos agrupados bajo el nombre PASCAL. Asi pues, de acuerdo con la definicion, PASCAL es una rama del directorio que agrupa a tres archivos: PROG, PROG7 y TAREA1. En terminos simples, es como si PASCAL fuera el "apellido" de los tres archivos con "nombres" diferentes.

Cada nombre de archivo esta seguido por: el nombre del lenguaje en el cual esta escrito. El directorio se puede ver como un arbol con un maximo de once niveles y varias ramas en donde cada "hijo" lleva como apellido el(los) nombre(s) de todos sus antecesores. Para el ejemplo anterior, los tres archivos bajo el directorio PASCAL podrian representarse como sigue :

RM81

PASCAL FORTRAN1 FORTRAN2

PROG PROG7 TAREA1

Se puede observar que la clave del usuario (RM81 en este caso) engloba a todas las ramas del directorio, esta se inserta como primer apellido de cada archivo del usuario pero unicamente sirve para efectos de manejo interno del directorio, es decir, para identificar la biblioteca del usuario; el usuario no debe considerarlo.

Cada renglon constituye un nivel y cada trayectoria particular constituye una rama, la cual si llega hasta el nivel de las raices del arbol, maneja a un archivo individualmente, pero si solamente llega a uno de los niveles anteriores, entonces maneja a una familia.

Por ejemplo, si la rama solo llega hasta pascal, involucra el manejo de la familia "PASCAL" constituida por PROG, PROG7 y TAREA1 que en este caso son hermanos. Pero una familia no siempre tiene que manejar a los hijos inmediatos, puede manejar a todos los descendientes deseados del nodo hasta donde llego la rama, es decir, a los hijos inmediatos de este nodo, a los hijos inmediatos de estos hijos,...., a los hijos de los hijos de los hijos,... y todos estos constituyen en su conjunto una familia.

Para representar los distintos niveles en un directorio, la terminal utiliza puntos que preceden al nombre del archivo, asi en el ejemplo anterior :

(RM81)
. PASCAL
. . PROG : ALGOL

. PASCAL estaria en el segundo nivel y constituye una rama
. . PROG estaria en el tercer nivel.
(RM81) estaria en el primer nivel.

En ese mismo ejemplo se encuentran tambien dos archivos llamados FORTRAN1 y FORTRAN2. La diferencia en el manejo de estos dos archivos y el de los agrupados bajo PASCAL esta en que estos ultimos pueden tratarse como una unidad, ademas de que se pueden agrupar a los archivos de acuerdo a las caracteristicas que tengan en comun, como en el caso anterior en que los tres estan escritos en lenguaje PASCAL, a pesar de tener nombres distintos.

Para agrupar los archivos por niveles en el directorio, es decir para manejar una familia, simplemente se antecede al nombre del archivo el nombre de todos sus antecesores (nombre de los nodos que se encuentran en la trayectoria de la rama) separados por diagonales "/", es decir, constituiran los apellidos de la familia de archivos, para estos deberan tomarse en cuenta las reglas que los nombres de los archivos (identificadores de archivos) deben cumplir, a saber :

1.- Maximo 17 caracteres por nombre.

2.- No incluir caracteres especiales.

3.- Tener un maximo de once nombres separados por diagonales (o sea un maximo de once niveles).

4.- El primer caracter de cada nombre debe ser alfabetico.

Es asi que los siguientes son identificadores validos para archivos :

JUAN/PASCAL/PROG1

PUC/DATOS

mientras que los siguientes no lo son :

JUAN/ALGOL/1

$/PROGRAMA2

Al crear un archivo se debera tomar en cuenta que no es conveniente usar demasiados niveles si estos no son necesarios, como en el siguiente ejemplo :

```
: P/A/S/C/A/L/EJEMPLO
#WORKFILE P/A/S/C/A/L/EJEMPLO: SEQ, 1 RECORD, SAVED
L
100 ESTE ES UN EJEMPLO DE ABUSO DE DIRECTORIOS
#
FILES P/A/S/C/A/L
(MCSI) ON PACK
  P
    A
      S
        C
          A
            L
              EJEMPLO : SEQDATA
#
```


Algunas de las ventajas que nos permite el uso de los niveles en los directorios al crear los archivos son :

Se puede ver el contenido de toda una familia de directorio con una sola instruccion :

```
FILES PASCAL
(MCSI) ON PACK
  PASCAL
    PROG : ALGOL
    PROG7 : ALGOL
    TAREA1 : ALGOL
#
```

o mandar escribir todos los archivos que pertenecen a una determinada rama del directorio poniendo el simbolo "/*" despues del nodo hasta donde llega la rama del directorio que se desea escribir para indicar que se refiere a toda una familia del directorio.

```
W EJEMPLO/*
# RUNNING 8408
#
```

o bien se pueden remover todos los archivos que pertenezcan a una rama en especial, por ejemplo :

```
REM EJEMPLO/=
# 5 FILES IN (MC81)EJEMPLO/= REMOVED ON PACK
FILES EJEMPLO
#NO FILE(S) ON PACK
#
```

en donde todos los archivos que pertenecian a la rama (familia) EJEMPLO fueron removidos (5 archivos en total).

NOTA: Aqui se emplea "rama" como sinonimo de familia, en donde una familia la integran los nodos que tienen un antecesor comun, pudiendose formar familias a partir del padre, abuelo, bisabuelo, etc.; es decir, un directorio es en si una familia pero integrada a la vez por una serie de familias (subfamilias).

En la seccion de comandos a veces se emplea directorio como sinonimo de familia (subdirectorio).

F) Tipos de archivos.

Todo archivo, ademas de pertenecer a alguna de las dos clases definidas con anterioridad (archivo de trabajo y archivo de biblioteca), debe poseer algun tipo.

El tipo se refiere (como la palabra lo indica) a la especie de datos que constituyen el archivo.

El tipo puede ser cualquiera de los que se enlistan en la siguiente hoja :

e	++ COBOL +++++
e	++ ALGOL +++++
e	++ BASIC +++++
e	++ FORTRAN +++
	++ PL/1 ++++++
	++ XALGOL ++++
	++ NOL +++++++
	++ DCF +++++++
	++ XFORTRAN ++
	++ BINDER ++++
	++ BASOL +++++
	++ GUAPO +++++
	++ DCALGOL +++
	++ DMALGOL +++
	++ ESPOL +++++
e	++ JOB +++++++
e	++ SEQ +++++++
e	++ DATA ++++++
	++ CDATA +++++

e indica que son los tipos mas comunmente utilizados.

G) TIEMPOS DE MAQUINA.

Son los tiempos que la maquina utiliza para realizar una tarea, y son:

ET.- Iniciales de "elapsed time" . Es el tiempo transcurrido desde que se inicia una tarea.

PT.- Iniciales de "process time" . Es el tiempo utilizado por el procesador de la maquina.

IO.- Iniciales de "input-output" . Es el tiempo utilizado por los dispositivos de entrada y salida de la maquina.

H) ARCHIVO FUENTE.

Es el archivo donde se encuentra el programa que el usuario tecleo y que esta en lenguaje de alto nivel (fortran, algol, etc).

I) ARCHIVO OBJETO.

Es el archivo que resulta de traducir el programa fuente al lenguaje que la maquina entiende (comando COMPILE) y solo este puede ser ejecutable (comando RUN) pues es el que esta en el idioma propio de la computadora.

SECCION V

V) OBSERVACIONES.

+Mas de un comando puede ser introducido en una misma linea siempre y cuando esten serarados entre si por " : " .

+Sin embargo hay excepciones, es decir, comandos que deben aparecer solos en una linea, o sea que no poseen la propiedad anteriormente mencionada. A continuacion se listan dichos comandos:

BYE
COPY
FIX
HELLO
SEQ
TAPE
WFL
DEL

•El sistema no aceptara comandos de CANDE hasta que el usuario sea identificado y entre a sesion (excepto el comando HELLO que es usado durante el proceso de identificacion).

•El procedimiento que realiza la maquina internamente para efectuar el reconocimiento, consiste en que, una vez introducida la clave completa (usercode/password), la busca en un directorio para ver si es una identificacion valida, y efectua la asignacion de los recursos, de acuerdo al tipo de clave que sea.

•Una vez efectuado el reconocimiento (si este fue el caso) da sesion, hora y fecha.

(ver,seccion de Preparacion para utilizar CANDE)

•Para salir del modo de secuencia automatica (ver comando SEQ) el usuario debera dar un doble "return".

("return es una tecla")

•Generalmente cuando la maquina termina de enviar su respuesta o de ejecutar el comando que se le ordeno manda un " 0 " a la siguiente linea para indicar que ya esta lista para recibir otro comando o instruccion, hay que esperar a que termine de dar su respuesta (que aparezca el 0) para que se pueda volver a teclear nuevamente.

•Una vez que se ha terminado de teclear una linea (ya sea texto, comandos, datos, etc.), hay que apretar la tecla "return" para que esta sea alimentada al sistema.

•En la seccion de comandos, el comando aparece enmarcado y la parte subrayada del comando indica su abreviacion, es decir, no es necesario teclear todo el comando ya que se puede teclear unicamente su abreviacion.

•Lo que aparece entre picoparentesis (<>) en la seccion de comandos, representa lo que puede tomar diferentes valores, nombres o indicaciones segun el usuario lo establezca.

CARACTERES ESPECIALES

.
#
$
%
&
,
(
)
*
@
<
>
.
/
_
|

COMPILADORES ESTANDAR

ALGOL
BASIC
BINDER
COBOL
DCALGOL
DMALGOL
ESPOL
FORTRAN
NDL
PL/I
XALGOL
XFORTRAN

SECCION VI

35

VI) PREPARACION PARA UTILIZAR EL SISTEMA.

Este procedimiento es indispensable para poder hacer uso de CANDE, e implica el entrar al sistema y que este efectue el reconocimiento (logging-on).

A continuacion se daran los pasos necesarios:

1.- Encender la terminal.

2.- Teclear HELLO.

3.- El sistema responde y solicita el USERCODE del usuario.

4.- Este paso se puede hacer en dos partes o en una sola:

I) Cuando se hace en dos partes:

a) Teclear el usercode.

b) La maquina solicita el passuord.

c) La maquina escribe unos simbolos sin significado.

d) El usuario teclea el passuord (este quedara impreso sobre los simbolos sin significado, de manera que no se distingue)

II) Si por el contrario, se hace en una sola parte el procedimiento sera el siguiente:

a) El usuario teclea el usercode y el passuord separados por una diagonal, pero ambos quedan a la vista (esto es riesgoso pues otra persona aiena puede ver el passuord).

5.- La maquina responde, y entre sus respuestas dara la sesion y asignara la hora y la fecha.

En el siguiente ejemplo se ve un procedimiento en el cual el sistema no reconocio la identificacion secreta (passuord) del usuario.

Entonces la maquina vuelve a solicitar el usercode y posteriormente el passuord y asi sucesivamente, hasta que el sistema reconoce la clave del usuario. (Otro caso puede ser que se termine un cierto lapso de tiempo y el sistema aun no ha reconocido la identificacion del usuario).

```
        HELLO
        #66700:1266 CANDE 31.280;YOU ARE DESPFI1(24)
        #ENTER USERCODE PLEASE
        ENS1
        #ENTER PASSWORD PLEASE
[1]---> BBBBBBBBBBBCBBBBB
        #INVALID USERCODE/PASSWORD;ENTER  USERCODE PLEASE
        BYE
        #ENTER PASSWORD PLEASE.
[1]---> BBBBBBBBBBBBBBBBB
        #INVALID USERCODE/PASSWORD;ENTER USERCODE PLEASE
        MCS1
        #ENTER PASSWORD PLEASE
[1]---> BBBBBBBBBBBBBBBBB
        #DEFAULT PRINT DESTINATION=SITE
        #SEMINARIOS PARA MARZO Y ABRIL
        #SESION 0971 14:34:55 04/19/82
```

[1] : Son los simbolos sin significado que la maquina manda imprimir y aqui queda impreso el passuord que el usuario teclea, el cual no se distingue pues queda escrito sobre los caracteres antes impresos.

Ahora se vera otro ejemplo en el cual todo el procedimiento es correcto y en el que se usa la opcion i en el cuarto paso de la preparacion para usar el sistema.

```
HELLO
#66700:126S CANDE  31.280:YOU ARE DESPFII(24)
#ENTER USERCODE PLEASE
MBS7
#ENTER PASSWORD PLEASE
[1]---> BBBBBBBBBBBBBBB
#DEFAULT PRINT DESTINATION=SITE
SEMINARIOS PARA MARZO Y ABRIL
#SESION 0973  14:35:33  04/19/82
```

[1] : Sobre estos simbolos, sin significado fue tecleado el password.

Y por ultimo se vera un ejemplo en el que se use la opcion ii del cuarto paso:

```
HELLO
#ENTER USERCODE PLEASE
MBS7/PASS
#DEFAULT PRINT DESTINATION=SITE
SEMINARIOS PARA MARZO Y ABRIL
#SESION 0964  14:33:05  04/19/82
```

A: Es conveniente usar la opcion i en el 4o. paso ya que como se puede ver en el primer ejemplo, de esta forma se evita el que el password quede a la vista, pues se pierde al quedar sobre los simbolos sin significado que la maquina da como parte de su res-

VII) DIAGRAMA DE FLUJO.

Disculpa:

Este cuaderno de trabajo, como se comento anteriormente, esta hecho con el proposito de servir como guia introductoria a CANDE para aquellos que nunca lo han manejado, o que tienen poca practica en su uso. Es por esto que posee una caracteristica que lo distingue y que es el hecho de que los comandos no se van introduciendo en orden alfabetico, pues esto no indicaria nada acerca de los pasos a seguir al usuario, dicho de otra manera este no sabria ni por donde empezar, continuar o terminar una vez que se ha instalado frente a la terminal.

Asi pues, este cuaderno va introduciendo los comandos en orden que satisface una secuencia muy utilizada al hacer uso de CANDE, o sea, los usuarios que por primera vez hacen uso de la terminal generalmente lo hacen teniendo en mente una meta (correr un programa u obtener listado o corrida) la cual requiere de una cierta secuencia de pasos a seguir. Para conseguir la meta la secuencia es:

1.- El proceso de preparacion para poder hacer uso de CANDE.

2.- El proceso de edicion del texto o programa (introduccion de lineas).

3.- El proceso de correccion de errores, inserciones nuevas, borrar lineas, etc., en breve el proceso de actualizacion debido a errores.

4.- Proceso de compilacion.

5.- Proceso de ejecucion (corrida).

6.- Proceso de fin de sesion.

Sin embargo hay casos que no obedecen a esta secuencia y es por esto que se presenta un diagrama de flujo en el cual se puede apreciar que se puede ir por distintos caminos, segun los requerimientos del usuario. Pero siguiendo el camino elegido y guiandose por el diagrama, el usuario siempre sabra que hacer para llevar a cabo su objetivo consultando los comandos que dicho diagrama indique le sean requeridos.

A continuacion se presenta el diagrama de flujo mencionado y enseguida una explicacion global y superficial de la interpretacion de dicho diagrama.

No se pretende por el momento que se entienda la funcion de los comandos, sino simplemente tener una idea de los pasos que involucra el diagrama en su conjunto.

Una vez hecho esto se puede proceder a interpretar a detalle dicho diagrama y para esto es necesario ir consultando los diferentes comandos (seccion de comandos) y ubicarlos en el diagrama para comprender su funcion.

NOTA: A pesar de que en el diagrama de flujo se presentan los casos mas comunes de secuencias realizadas durante sesiones, muchos de los comandos que son listados en este cuaderno de trabajo, pueden ir en diferentes lugares tanto de las secuencias presentadas como de otras secuencias que aqui no se presentan, debido a que existe una inmensa variedad de diferentes secuencias realizadas durante sesion (en cada sesion el usuario hace diferentes acciones de acuerdo a sus necesidades)

Sin embargo una vez comprendidos los comandos y su funcion para los casos aqui expuestos, el lector estara en disponibilidad de hacer uso extensivo de ellos en diferentes secuencias que surjan de acuerdo a sus requerimientos.

Exrlicacion general del diagrama de flujo.

Antes que nada cabe mencionar que lo primero con lo que el usuario debe contar son dos cosas:

-Estar en una terminal activa y encendida.

-Poseer una clave (USERCODE/PASSWORD) valida para el sistema.

Si lo anterior es satisfecho entonces se procede a efectuar el proceso de preparacion para poder hacer uso de CANDE:

a) Saludar a la terminal mediante el comando HELLO e identificarse por medio de la clave para que esta sea reconocida y, si se desea obtener un listado del archivo en que se trabajara o bien los resultados de la corrida por impresora, se podra cargar la sesion mediante los comandos CHARGE, SPLIT; este ultimo fijara el encabezado de la impresion.

b) Es probable que al entrar en sesion aparezca en la pantalla un mensaje diciendo que hay archivos en RECOVERY (RECOVERY FILES) debido a una falla del sistema en una sesion, en este caso se usaran las opciones RECOVERY o DISCARD segun se desee recobrar o eliminar dicho(s) archivo(s).

c) Para obtener la lista de archivos (directorio) que existen en la biblioteca del usuario, se usara el comando FILES. Esto es con el objeto de no bautizar un archivo que se va a crear con el nombre de uno ya existente, o bien para efectos de consulta.

d) Se usara el comando MAKE para solicitar la creacion de un nuevo archivo y posteriormente el comando SEQ que dara la secuencia automatica deseada para editar el archivo. Durante el tecleado del nuevo archivo, se podran corregir errores o introducir margenes sin necesidad de salirse de secuencia mediante los comandos FIX, DEL , MARGIN o TAB (con todas sus opciones). Una vez terminado se guardara el archivo en disco mediante el comando SAVE para que pase a formar parte de la biblioteca del usuario.

Si el archivo en el que se desea trabajar ya esta creado, se mandara traer de la biblioteca mediante el comando GET haciendo que ese archivo pase a ser el archivo propio de trabajo y poder:

d.1) Listarlo: Mediante WRITE o LIST dependiendo si se desea el listado por impresora o por terminal.

d.2) Actualizarlo: Mediante las opciones FIX, DELETE, INSERT, MOVE, FIND, REPLACE, RESEQ, UPDATE, etc. (proceso de correccion)

d.3) En caso de no existir aun el correspondiente programa objeto, habra que compilarlo mediante el comando COMPILE, y en caso de no existir errores se podra proseguir a su ejecucion.

Si el archivo tuvo errores de sintaxis sera necesario volver a actualizarlo mediante los comandos mencionados en el inciso d.2) para posteriormente volver a compilar.

Durante la compilacion se podra preguntar al sistema por medio del comando de control ?CS , el estado de la compilacion.

e) Si el programa ya compilado desea ejecutarse podran usarse los comandos RUN o RUN:(ecuacion de archivo). Este ultimo se usara si los resultados se desean obtener en algun otro dispositivo que no sea terminal, por ejemplo impresora, disco, etc. . Si durante la ejecucion es necesario suministrar datos, estos se daran en el momento en que la terminal los pida. Durante este proceso de ejecucion se podra preguntar por el status de la corrida mediante los comandos de control ?STA, ?TI, ?Y o bien descontinuarla mediante ?DS. Una vez terminada la ejecucion se podra usar el comando BACK para revisar los resultados si es que estos iban a imprimirse en papel. .

f) Una vez que se ha terminado de trabajar con ese archivo se dara SAVE o REMOVE para guardarlo o removerlo. En caso de ser guardado se le podra cambiar la seguridad mediante el comando SECURITY.

g) Si se desea iniciar una nueva sesion se usara el comando SPLIT, de lo contrario se usara BYE para terminar la sesion

NOTA: Los incisos indicados en esta breve exlicacion corresponden a los indicadores marcados en el propio diagrama de flujo.

VIII) COMANDOS.

En esta seccion se describen tanto los comandos de accion como los de control de CANDE.

Los comandos van en el orden de la secuencia que se dio en la seccion del diagrama de flujo, excepto algunos comandos que van en el apendice debido a que pueden entrar en cualquier lugar de la secuencia, es decir, se pueden utilizar en el momento deseado.

Existen 2 tipos de comandos, que son:

A) Los comandos de accion:

Son los que sirven para iniciacion de tareas, creacion de archivos, manipulacion de la libreria del usuario, edicion de textos, etc.

B) Los comandos de control:

Estos comandos proveen al usuario la habilidad para controlar e interrogar el medio operativo, deben estar precedidos por el simbolo " ? " .

Tambien en esta seccion se muestran algunos de los errores mas comunes, dandose a la vez una explicacion del porque de su ocurrencia y la forma de corregirlos.

Se tiene a continuacion la presentacion de cada de los comandos.

NOTA: Los ejemplos que se presentan para las diversas opciones de los diferentes comandos, son los que los alumnos universitarios, becarios del Programa de Formacion de Recursos Humanos (P.U.C.), utilizaron para practicar y comprender en forma clara los comandos.

Ademas constituyen un dialogo exacto ya que son listados de programas realizados durante sesiones llevadas a cabo en teleti-po.

Se hace uso de un mismo texto para hacer la ejemplificacion en los diferentes comandos de actualizacion con el fin de que el lector se familiarice con el y comprenda mas facilmente los comandos viendo la evolucion que va sufriendo dicho texto.

Para la compilacion y la corrida se usan unos programitas que solo leen variables y las imprimen ya que el texto usado en los demas comandos no se puede compilar ni correr.

NOTA DEL APENDICE: Algunos comandos no tienen un lugar especifico para la secuencia aqui tratada, es decir, los podemos insertar en cualquier parte del diagrama de flujo. Estos comandos aparecen explicados en la seccion del apendice.

HELLO

Es el primer comando que el usuario debe dar cada vez que requiera hacer uso de la terminal. Hay que saludarla antes que nada!! (Ver la seccion VI "preparacion para hacer uso de CANDE")

CHARGE

Este comando es utilizado para identificar las tareas realizadas en la sesion y que se obtienen por medio de una impresora, a traves de un listado, este listado puede ser producto de una corrida, compilacion u otra tarea.

OPCIONES:

1) CHARGE <Identificador>

OPCION 1:

CHARGE <Identificador>

Para esta opcion, el identificador o string (cuerda), puede estar encerrada entre comillas. Si el identificador es mas grande de 10 caracteres, solo los primeros 10 son usados. El nombre que se le asigne al identificador es el que aparecera como titulo en la caratula del listado.

Ejemplo:

```
[1]---> CHARGE "CARGA"
[2]---> #SESION 3043  ET=7:41.9  PT=0.6  IO=0.2
[3]---> #CONTINUE SESSION 3043 14:26:08  04/23/82
```

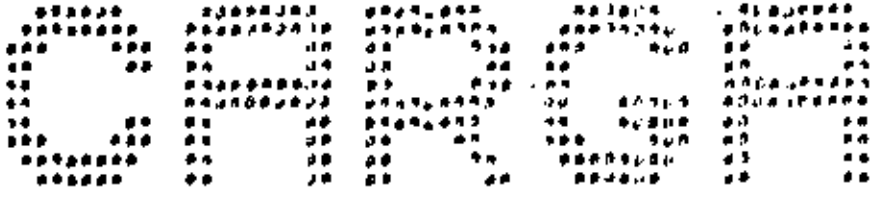
[1] : Se hizo un CHARGE, donde el identificador es CARGA, notar que esta entre comillas. Si el identificador es mas grande de 10 caracteres, solo los primeros 10 son significativos.

[2] : La maquina contesta con el numero de sesion que se tiene en ese momento y tres tiempos.

[3] : Por ultimo la maquina indica que la sesion continua y - provee la hora y la fecha.

El resultado de hacer esto se obtiene al ver el listado, c robando que efectivamente lleva el nombre que le fue asignado al identificador.

A continuacion se muestra el titulo que lleva el listado, debido al uso del comando.



MAKE

Este comando tiene por objeto crear un archivo de trabajo, si anteriormente se tiene creado otro archivo de trabajo, este comando sera rechazado por la maquina, indicando que antes de hacer esta operacion, se tiene que salvar o remover el archivo anterior, esto ultimo se hace por medio de los comandos SAVE o REMOVE.

El usar el comando implica darle a nuestro archivo de trabajo un nombre, que puede estar constituido por uno o mas identificadores, en donde cada uno puede tener hasta 17 caracteres y cada identificador debe estar separado por una diagonal (/). El nombre utilizado para el archivo debe empezar con un caracter alfabetico. Ademas se debe especificar de que tipo es el archivo de trabajo.

Caracteristicas de un Archivo de Trabajo
- Es uno y solo uno
- Puede ser modificada su informacion.
- Se puede tener presente mediante los comandos GET, LOAD o MAKE.

A continuacion se muestran tipos de archivos de trabajo que se pueden crear.

Tipos de Archivos de Trabajo
- Datos (Contienen Informacion)
  - SEQ (SECUENCIAL)
  - DATA
  - CHARACTER SEQ
  - CHARACTER DATA
- Programas (Proposicion)
  - ALGOL
  - BASIC
  - FORTRAN
  - PL/I
  - PASCAL
  - COBOL

Son los tipos mas comunmente utilizados, para consultas sobre otros tipos existentes pero menos usados, consultar en la seccion de terminologias lo concerniente a tipos de archivos.

OPCIONES:

1) MAKE <Nombre o Identificador>

2) MAKE <Nombre o Identificador> <Tipo>

OPCION 1:

MAKE <Nombre o Identificador>

Para la opcion uno, se utiliza el comando y un nombre constituido por uno o mas identificadores separdos por una diagonal (/). En esta opcion no se definio el tipo, entonces es dado por default el tipo SEQ (SECUENCIAL).

Ejemplo:

```
                #
[1]---> MAKE PASCALE
[2]---> #WORKFILE PASCALE: SEQ
```

[1] : Se creo un archivo cuyo nombre o identificador es PASCALE, notar que no se especifica de que tipo es este archivo.

[2] : La maquina responde que es un archivo de trabajo y le asigna el tipo SEQ (SECUENCIAL).

OPCION 2:

MAKE <Nombre o Identificador> <Tipo>

En la segunda opcion, a diferencia de la primera si es definido el tipo a utilizar, que puede ser alguno de los tipos existentes:

Ejemplo:

```
                #
[1]---> MAKE EJEMPLO/CANDE DATA
[2]---> #WORKFILE EJEMPLO/CANDE: DATA
```

[1] : En este ejemplo se ha definido el nombre del archivo con mas de un identificador separados por una diagonal (EJEMPLO/CANDE) y se ha utilizado el tipo DATA.

[2] : La maquina responde que se trata de un archivo de trabajo que se llama EJEMPLO/CANDE y que es de tipo DATA. El haber elegido el tipo DATA no implica que todos los archivos de trabajo sean de este tipo necesariamente, se puede hacer uso de los diferentes tipos especificados.

Si se crea un archivo de trabajo con el mismo nombre con el cual otro fue creado y almacenado en biblioteca anteriormente, la maquina responde que ese archivo ya existe y se encuentra almacenado, de modo que se puede trabajar con el si asi se desea.

Ejemplo:

```
                #
[1]---> MAKE PASCAL ALGOL
        #WORKFILE PASCAL: ALGOL
        SEQ100+10
        100 PROGRAM REPITE;
        110 VAR
        120   I,NUM:INTEGER;
        130 BEGIN
        140 FOR I:=1 TO 10 DO BEGIN
        150 READ(NUM);WRITE(NUM) END
        160 END.
        170
        #
[2]---> C W PASCAL;R
        #UPDATING
        #WAITING FOR AVAILABLE TASK
        #COMPILING 9490
        #ET=6.2 PT=1.2 IO=1.3
[3]---> SA
        #WORKFILE PASCAL SAVED
[4]---> MAKE PASCAL ALGOL
        #SOURCE AND OBJECT FILE ALREADY PRESENT
```

[1] : Se crea un archivo que se llama PASCAL y es de tipo ALGOL.

[2] : Se compila y con esto se crea un programa objeto ya que la compilacion fue ejecutada.

[3] : Es salvado mediante el comando SAVE y asi es almacenado en biblioteca.

[4] : Posteriormente se pretende crear un archivo con el mismo nombre y del mismo tipo, a lo cual la maquina responde que ese archivo ya existe (o sea tanto el archivo fuente como el objeto) y esta listo para ser usado.

SEQ

Este comando genera automaticamente numeracion en las lineas de un archivo de trabajo, es decir, secuencia automatica.

Una de las ventajas que se presentan al utilizar el comando, es el insertar una linea o un conjunto de ellas, entre dos lineas del archivo, para ello se debe elegir una secuencia de acuerdo a lo necesitado.

OPCIONES:

1) SEQ <Base> + <Incremento>

2) SEQ NEXT + <Incremento>

3) SEQ END + <Incremento>

4) Fuera de Secuencia

OPCION 1:

SEQ <Base> + <Incremento>

En la primera opcion, la base especifica el numero de secuencia asignado a la primera linea del archivo, el incremento es la variacion en la numeracion de las lineas subsecuentes. Con esta opcion se dice que se esta dando una secuencia. Si no se especifica por default es 100+100.

Ejemplo:

```
      #
[1]---> MAKE PRUEBA/SEQ DATA
        #WORKFILE PRUEBA/SEQ DATA
[2]---> SEQ 10+5
[3]---> 10 UTILIDAD DEL COMANDO
        15 PARA FINES PRACTICOS
        20 VEREMOS LOS RESULTADOS DE UTILIZARLO
        25
[4]---> #
```

[1] : Se crea un archivo llamado PRUEBA/SEQ de tipo DATA.

[2] : Se especifica una secuencia determinada, utilizando la primera opcion, la base es el numero 10 y el incremento es de 5.

[3] : Como podra verse la base es igual al numero de secuencia de la primera linea del archivo y los incrementos subsecuentes son de 5 a partir de ese valor.

[4] : El signo "#" implica que ya se esta fuera de secuencia automatica y esto se logra oprimiendo dos veces la tecla "return".

OPCION 2:

SEQ NEXT + <Incremento>

Para esta opcion, en el NEXT quedara el valor numerico de la ultima linea del ultimo comando SEQ, MOVE, INSERT, RESEQ que se haya utilizado, si ninguno de estos comandos fue usado el 100 es asumido. El incremento sera la variacion de las lineas subsecuentes a partir de dicho valor.

Ejemplo:

```
      #
[1]---> SEQ NEXT+3
        25 INTRODUCIR UNA LINEA
[2]---> 28
        #
```

[1] : Notar que el valor asignado a NEXT es el de la ultima linea del archivo donde se utilizo el comando SEQ para este ejemplo.

[2] : La variacion de las siguientes lineas esta dada por el incremento.

OPCION 3:

SEQ END + <Incremento>

En la ultima opcion al END se le asigna como primer numero de secuencia al numero mas grande utilizado en el archivo, mas el incremento declarado.

Ejemplo:

```
      #
[1]---> SEQ END+4
        29 VERIFICANDO EL RESULTADO
        33 CHECADO
        37
        #
```

[1] : El valor asignado al END es el de la ultima linea del archivo mas el incremento especificado, esto es 25+4 o sea 29.

OPCION 4:

Al estar fuera de secuencia automatica, basta con teclear el numero de la linea deseada junto con su contenido.

Ejemplo:

```
        MAKE PRUEBA/SEQ DATA
        $WORKFILE PRUEBA/SEQ: DATA
        SEQ 10*5
        10 UTILIDAD DEL COMANDO
        15 PARA FINES DE COMPROBACION
        20 VEREMOS LOS RESULTADOS AL UTILIZARLO
        25 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
        30
        *
        LIB
        10 UTILIDAD DEL COMANDO
        *
[2]---> 10 INTRODUJE LINEA FUERA DE SECUENCIA
        LIB
[1]---> 10  INTRODUJE LINEA FUERA DE SECUENCIA
        *
```

[1] : Al reteir el numero de una linea de la secuencia, el contenido anterior se borra y queda el nuevo.

[2] : Del ejemplo anterior se puede apreciar que es posible introducir lineas dentro de la secuencia tecleando directamente el numero de linea que se desea cambiar.

Ejemplo de secuencia automatica con valores dados por default:

En este ejemplo se editara el texto con errores que se utilizaran en la parte de actualizacion.

```
MAKE TEXTO/III SEQDATA/S
$WORKFILE TEXTO/III: SEQ
100 ..........................................
200 ##########################################
300
400 ..........................................
500 ##########################################
600          EL PERRO Y SU AMIGO
700 SOBRE EL ANCHO PUENTE DE UN RIO
800 PASABA UNA POBRE PERRO.
900 LLEVANDO UNA HUESO EN EL HOCICO
1000 MIRANDO HASTA AVAJO,VIO SU PROPIA AMIGO
1100 REFLEJADO EN EL AGUA NEGRA
1200 Y DIJO PERA SI:
1300
1400      "HOLA"
1500      $$$$$
1600      "HOLA"
1700 EL CUANTO MEJOR QUE EL MIO DEBIERA
1800 SER EL HUESO QUE YEVA ESE
1900 AFORTUNADO COMPANERO !!
2000 Y CAYO AL FONDO DEL RIO QUEDANDOSE
2100 EL POBLE SIN NINGUNO
2200 Y TRATO DE ARREVATARSELO.
2300 MAS AL ABRIR LA BOCA,EL HUESO
2400 ESCAPO DE ENTRE SUS DIENTIS
```

MARGIN

Este comando se utiliza para el control del margen de una linea o de todo el texto introducido en nuestro archivo de trabajo.

El caracter @ debe ser utilizado en lugar de MARGIN si la secuencia esta dada automaticamente (comando SEQ).

OPCIONES:

1) MARGIN + <Numero de Columnas>

2) MARGIN - <Numero de Columnas>

3) MARGIN ?

4) @ + <Numero de Columnas> : <Texto>

OPCION 1:

MARGIN + <Numero de Columnas>

En la primera opcion se hace uso del comando para dar margenes hacia la derecha, el margen sera tan grande como lo indique el numero de columnas.

Ejemplo:

```
      #
      MAKE MARGENES DATA
      #WORKFILE MARGENES: DATA
[1]---> MARGIN+10
      #
      SEQ10+5
      100*10: LUGAR DE IMPRESION
      15 DEBE DE EFECTUARSE EL COMANDO
      20 ESPERANDO EL RESULTADO FINAL AL SER LISTADO
      25 .....
      30
      #
[2]---> LIST
      10                    LUGAR DE IMPRESION
      15          DEBE DE EFECTUARSE EL COMANDO
      20          ESPERANDO EL RESULTADO FINAL AL SER LISTADO
      25          .....
      #
```

[1] : Se utilizo el comando fuera de secuencia, lo cual dara un margen de 10 posiciones al texto introducido y la maquina responde con un "#" lo cual quiere decir que se ejecutara esta accion.

[2] : Para comprobar el resultado de utilizar esta opcion, se sale de secuencia y se lista el archivo creado. Al dar el LIST se comprueba que fue ejecutada la opcion, observando que existe un margen de 10 posiciones para todo el texto.

OPCION 2:

MARGIN - <Numero de Columnas>

En esta opcion se utiliza este comando para decrementar el tamaño del margen, el margen disminuira tantas posiciones como lo indique el numero de columnas despues del signo. Se aplica en la misma forma que la opcion anterior.

OPCION 3:

MARGIN ?

Aqui el caracter ? precedido del comando se usa para verificar como se esta utilizando el comando MARGIN.

Ejemplo:

```
        #
        MAKE COMPRUEBA/MARGIN DATA
        #MODIFILE COMPRUEBA/MARGIN: DATA
        MARGIN+10
        #
        SE@10+1
        10@+7: USO DEL MARGIN
        11 COMPROBACION DE LA TERCERA OPCION
        12 Y UTILIDAD DE LA MISMA
        13 ....................................
        14
        #
[1]---> MARGIN ?
[2]---> #MARGIN @11
```

[1] : Se consulta el MARGIN que se encuentra en ese momento vigente y como el uso de la opcion 4 solo se refiere a la linea 10, el ultimo margin dentro del texto es el que tiene la linea 13, que es un margen de 10 posiciones.

[2] : Por lo anterior la maquina contesta que el margen es la columna que precede a la numero 11 (MARGIN @11).

OPCION 4:

@ + <Numero de Columnas> : <Texto>

Se debe definir el numero de columnas que se desea mover el texto, el cual tambien debe estar definido dentro de la misma opcion. Cuando esta dentro de secuencia automatica, en lugar de

utilizar la palabra "MARGIN", esta se sustituye por el simbolo "@", en este caso este simbolo debera ir inmediatamente despues de el numero marcado por la secuencia .

Ejemplo:

```
        #
        MAKE MARGENES DATA
        #MODIFILE MARGENES: DATA
        MARGIN+10
        #
[1]---> $E@10+5
[2]---> 10@+10: LUGAR DE IMPRESION
        15 DEBE DE EFECTUARSE EL COMANDO
        20 ESPERANDO EL RESULTADO FINAL AL SER LISTADO
        25 ....................................
        30
        #
        LIST
[3]---> 10          LUGAR DE IMPRESION
        15      DEBE DE EFECTUARSE EL COMANDO
        20      ESPERANDO EL RESULTADO FINAL AL SER LISTADO
        25      ....................................
        #
```

[1] : Se pide secuencia automatica, por esto si se usa MARGIN debe de hacerse con @.

[2] : Se especifica el MARGIN a utilizar en esa linea (con @).

[3] : Al dar un LIST se observa el resultado, que es un margen de 20 posiciones para la linea numero 10, la pregunta sera por que 20 ? esto es porque el margen es de 10 posiciones para todo el texto y 10 que se especificaron para la linea numero 10. Se hace notar que en la edicion aparece @+10, pero posteriormente al momento de listarlo no es asi (no forma parte del texto).

TAB

Este comando nos permite definir un caracter de tabulacion.

OPCIONES:

1) TAB = <Caracter Especial> <Entero>

2) TAB ?

3) TAB

OPCION 1:

TAB <Caracter Especial> <Entero>

En esta opcion existe una restriccion en la utilizacion del caracter especial, no se permite utilizar los caracteres ? * null. El entero especifica el numero de columna donde se va a poner el margen cuando se encuentre el caracter especificado por el comando TAB.

Ejemplo:

```
      LIST
      10                LUGAR DE IMPRESION
      15          DEBE DE EFECTUARSE EL COMANDO
      20          ESPERANDO EL RESULTADO FINAL AL SER LISTADO
      25          ..........................................
      #
[1]---> TAB=$ 25
      #
[2]---> 24$ FIN
      #
      L
      10                LUGAR DE IMPRESION
      15          DEBE DE EFECTUARSE EL COMANDO
      20          ESPERANDO EL RESULTADO FINAL AL SER LISTADO
      24                FIN
      25          ..........................................
      #
```

[1] : Se declaro un TAB especifico en donde se tomo como caracter especial el signo "$" y como entero el numero 25 y la palabra FIN.

[2] : Se inserto una linea en el archivo utilizando el caracter especial antes especificado para el TAB a utilizar. Al listar el archivo se observa que el comando TAB fue ejecutado al colocar la palabra FIN despues de la columna 25.

En este ejemplo se necesito el caracter especial ya que este es el que se busca en las lineas que se inserten, para saber a partir de donde dejara los espacios. Es por esto que se permite tener varios tabs declarados, siempre y cuando tenga cada uno su propio caracter especial, este ultimo es un indicador.

Tambien es facil notar que al editar la linea es preciso poner dicho caracter (el cual puede ir en cualquier parte de la linea) en el lugar donde se requiera dejar los blancos, pero al

listar veaos que no aparece este (no es parte del texto).

OPCION 2:

TAB ?

Esta opcion permite saber como y en donde se esta utilizando el TAB, el utilizar esta opcion para el TAB es solo para efectos de consulta.

Ejemplo:

```
      LIST
      10                LUGAR DE IMPRESION
      15          DEBE DE EFECTUARSE EL COMANDO
      20          ESPERANDO EL RESULTADO FINAL AL SER LISTADO
      25          ........................................
      #
[1]---> TAB=# 25
      #
      24# FIN
      #
[2]---> TAB?
      #TAB = # 25
```

[1] : Se declaro un TAB especifico, el caracter especial es "#" y el entero es 25.

[2] : Se utiliza la segunda opcion para saber que TAB esta vigente, como el ultimo TAB utilizado fue el descrito anteriormente solo despleso el mismo TAB.

OPCION 3:

TAB

Aqui un comando TAB sin parametros removera todos los tabs que esten declarados.

Ejemplo:

```
      #
[1]---> TAB
[2]---> #TABS CLEARED
```

[1] : Se utiliza el comando TAB sin parametros

[2] : Aqui la maquina responde que todos los tabs declarados son borrados.

GET

El comando GET y el comando LOAD son sinonimos.

Este comando tiene por objeto traer presente un archivo de biblioteca (es.decir un archivo que con anterioridad fue almacenado utilizando el comando SAVE) para convertirlo en un archivo de trabajo.

El comando no es aceptado por la maquina si anteriormente en la sesion se tenia un archivo, que aun no ha sido salvado o removido y por lo tanto este ultimo costituye aun el archivo de trabajo utilizado.

OPCIONES:

1) GET <Nombre o Identificador>

2) GET <Nombre o Identificador> AS <Nombre o Identificador>

OPCION 1:

GET <Nombre o Identificador>

En esta opcion se utiliza el comando para traer presente un archivo que fue creado y almacenado en biblioteca con anterioridad.. El nombre o identificador debe ser exactamente el mismo con el cual fue creado, de otra manera la maquina contestara que el archivo requerido no existe.

Ejemplo:

```
        #
[1]---> GET EJEMPLO/CANDE
[2]---> #INDEXFILE EJEMPLO/CANDE DATA, 4 RECORDS, SAVED
        LIST
        100 ESTE ARCHIVO SE CREO PARA
        200 EJEMPLIFICAR DIVERSOS COMANDOS
        300 UTILIZADOS DENTRO DE ESTE MANUAL
        400 ..............................
        #
```

[1] : Se hace uso del comando y el nombre con el cual el archivo fue creado (en este caso EJEMPLO/CANDE).

[2] : La maquina contesta que si existe ese archivo, que es un archivo de trabajo de tipo DATA y ademas contiene 4 registros. por ultimo dice que fue anteriormente salvado.

+++ El comando GET crea una copia del archivo de biblioteca solicitado y esta copia es la que constituye el archivo de trabajo, es decir, el original no es afectado a menos de que el usuario decida darle un SAVE a esta copia, destruyendo asi la version anterior residente en biblioteca.

OPCION 2:

GET <Nombre o Identificador> AS <Nombre o Identificador>

Para esta opcion se utilizan dos nombres de archivo, el primero se refiere al nombre con el cual fue creado originalmente y el segundo es el nombre con el cual se va a manejar en ese momento.

El tener presente de esta manera un archivo permite no so-

dificar el original y crear otra version a partir del original, pero esta version constituira un archivo con otro noubre.

Ejeario:

```
[1]---> MAKE MARGENES DATA
        #WORKFILE MARGENES; DATA
        MARGIN+10
        SEQ10+5
        102+10 LUGAR DE IMPRESION
        15 DEBE DE EFECTUARSE EL COMANDO
        20 ESPERANDO EL RESULTADO FINAL AL SER LISTADO
        25 ..........................................
        30
        #
        SAVE
        #UPDATING
        #WORKSOURCE MARGENES SAVED
[2]---> GET MARGENES AS MARGENES2;LIST
        #WORKFILE MARGENES2;SOURCE IS (MCS1)MARGENES ON PACK;DATA 4 R
        10                LUGAR DE IMPRESION
        15          DEBE DE EFECTUARSE EL COMANDO
        20          ESPERANDO EL RESULTADO FINAL AL SER LISTADO
        25          ..........................................
        #
```

[1] : En el ejeario anterior se creo un archivo de trabajo con el noubre de MARGENES, el cual fue salvado y almacenado.

[2] : Utilizando la segunda orcion para tenerlo presente pero bajo el noubre de MARGENES2 y es listado, la maquina contesta que MARGENES2 es un archivo de trabajo pero que el fuente es MARGENES, que tambien esta en PACK, es de tipo data y contiene 4 registros.

En ese momento se esta trabajando con un archivo que se llama MARGENES2, que contiene la misma informacion que existe en el archivo MARGENES, pero este ultimo archivo no sera alterado con las modificaciones que sean hechas sobre MARGENES2 aunque este sea salvado, pues queda almacenado bajo el noubre de MARGENES2.

Esta parte es la que ilustrara el mal uso del comando:

Ejeario:

```
[1]---> MAKE EJEMPLO/CANDE DATA
        #WORKFILE EJEMPLO/CANDE; DATA
        SEQ100+100
        100 ESTE ARCHIVO SE CREO PARA
        200 EJEMPLIFICAR DIVERSOS COMANDOS
        300 UTILIZADOS EN ESTE MANUAL
        400 ..........................
        500
        #
        G EJEMPLO/CADE
        #REMOVE OR SAVE WORKFILE
[2]---> SA
        #UPDATING
        #WORKSOURCE EJEMPLO/CANDE SAVED
[3]---> GET EJEMPLO/CADE
[4]---> #NO FILE:EJEMPLO/CADE
```

[1] : Se crea un archivo con un noubre especifico.

[2] : Es salvado, almacenandolo de esta manera en biblioteca.

[3] : Se trata de traerlo presente, pero no se utiliza el noubre con el cual fue creado originalmente (omision de la letra N).

[4] : La maquina responde que no encontro ningun archivo bajo ese noubre, es decir lo busco en biblioteca con ese noubre y no lo encontro.

LIST

El comando LIST y el comando PRINT son sinonimos.

Este comando tiene por objeto listar el contenido parcial o total de un archivo.

OPCIONES:

1) LIST <Nombre o Identificador>

2) LIST <Numero de Secuencia de una Linea>

3) LIST <Un Determinado Rango de la Secuencia>

4) LIST *

5) LIST <Numero de Secuencia de una Linea> - <END>

OPCION 1:

LIST <Nombre o Identificador>

Aqui el uso del comando es para listar el contenido total de un archivo, si el nombre o identificador no es especificado el archivo de trabajo es asumido, de lo contrario se trata de un archivo de biblioteca.

Ejemplo:

```
[1]---> LIST EJEMPLO/LIST
[2]---> #FILE (NCSI)EJEMPLO/LIST ON PACK
[3]---> 100 EJEMPLIFICACION DEL COMANDO LIST
        200 ESTE COMANDO SE UTILIZA PARA LISTAR EL
        300 CONTENIDO TOTAL O PARCIAL DE UN ARCHIVO
        400
        500
        600 LAS DIFERENTES MANERAS DE USARLO SERAN
        700 DADAS POR CADA UNA DE LAS OPCIONES QUE
        800 TIENE EL COMANDO, ASI COMO EJEMPLOS QUE
        900 DEMUESTREN SU UTILIDAD.
        #
```

[1] : En el ejemplo anterior se hizo uso de la primera opcion al usar el comando y el nombre del archivo (notar que no se dio un GET. EJEMPLO/LIST), es por esto que para listar el archivo se especifico cual archivo se requeria, ya que no se trata de un archivo de trabajo sino de uno de biblioteca.

[2] : La maquina contesta que se trata de un archivo que esta bajo, la clave que enmarcan los parentesis y ademas esta en PACK.

[3] : Como consecuencia del uso del comando, aparece totalmente el contenido del archivo.

OPCION 2:

LIST <Numero de Secuencia de una Linea>

Aqui el uso de esta opcion es necesario para listar una linea en particular de nuestro archivo (lista la linea indicada).

ejemplo:

```
      #
]---> L600
]---> 600 LAS DIFERENTES MANERAS DE USARLO SERAN
      #
```

] : Se hace uso de la segunda opcion, queriendo listar una linea en particular de un archivo (en este caso el archivo se encuentra presente).

2] : Aparece la linea que se quiso listar.

OPCION 3:

LIST (Un Determinado Rango de la Secuencia)

El comando es utilizado para listar el contenido parcial de un archivo, esto es un bloque en especial la unica restriccion es que este bloque se encuentre contenido dentro del archivo (lista de la linea X a la linea Y donde X < Y).

Ejemplo:

```
      #
E1]---> LIST 100-400
E2]---> 100 EJEMPLIFICACION DEL COMANDO LIST
        200 ESTE COMANDO SE UTILIZA PARA LISTAR EL
        300 CONTENIDO PARCIAL O TOTAL DE UN ARCHIVO
        400
        #
```

E1] : En este ejemplo se considera la tercera opcion, se listara un bloque determinado de un archivo que en este caso esta presente, aqui se desea listar de la linea 100 a la linea 400.

E2] : La accion es ejecutada al aparecer el bloque, esto es de la linea 100 a la 400 inclusive.

OPCION 4:

LIST =

El uso de la cuarta opcion hace referencia a la ultima linea utilizada anteriormente, su objeto es verificar alguna accion ejecutada sobre esta.

Ejemplo:

```
      #
E1]---> F100/LIST/LIST:
      #
E2]---> L=
        100 EJEMPLIFICACION DEL COMANDO LIST:
        #
```

E1] : Se hace una modificacion en la linea 100 del archivo de trabajo, el cambio es ejecutado al aparecer el signo "#".

E2] : Se usa la cuarta opcion (L=), y la linea en donde se realizo el cambio aparece listada. Como se dijo anteriormente esta opcion se utilizo para verificacion.

OPCION 5:

LIST (Numero de Secuencia de una Linea) - END

Por ultimo esta opcion listara una parte de un archivo , a diferencia de la tercera opcion, esta tendra como ultima linea el final del archivo.

Ejemplo:

```
        #
[1]---> LIST700-END
        700 DADAS POR CADA UNA DE LAS OPCIONES QUE
        800 TIENE EL COMANDO, ASI COMO EJEMPLOS QUE
        900 DEMUESTREN SU UTILIDAD.
[2]---> 1000 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
        #
```

[1] : Se desea listar de la linea 700 al final (END) haciendo uso de la ultima opcion.

[2] : Aqui se puede observar la opcion lista hasta la ultima linea del archivo.

WRITE

Este comando actua sobre un archivo de trabajo o de biblioteca.

Es permitido imprimir un archivo o parte de el.

OPCIONES:

1) WRITE <Nombre o Identificador> (Archivo de Biblioteca)

   WRITE (Por Default Archivo de Trabajo)

2) WRITE <Nombre o Identificador> :DOUBLE (Archivo de Biblioteca)

   WRITE:DOUBLE (Por Default Archivo de Trabajo)

OPCION 1:

WRITE <Nombre o Identificador>

Esta opcion tiene por objeto imprimir un archivo, el dispositivo de impresion por default es impresora, o sea, al realizar un WRITE el resultado se obtiene en un listado.

Ejemplo:

```
        #
[1]---> WRITE EJEMPLO/WRITE
[2]---> #RUNNING 5993
        #
```

[1] : En el ejemplo se hace uso de la primera opcion para imprimir el contenido de un archivo llamado EJEMPLO/WRITE.

[2] : La maquina responde que el comando se esta ejecutando al

aparecer RUNNING y un numero asignado por la maquina automaticamente, resultado lo obtenemos al recoger el listado.

Aqui se muestra la parte del listado en donde se imprimio el archivo.

EJEMPLO/WRITE (04/16/82)

```
100 EJEMPLIFICACION DEL COMANDO WRITE
200 USADO PARA LA IMPRESION DE ARCHIVOS
300 LOS ARCHIVOS NO SON COMPILADOS
400
500 LAS OPCIONES ILUSTRARAN SU USO
600 @@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

OPCION 2:

WRITE <Nombre o Identificador> :DOUBLE

La opcion dos, a diferencia de la primera, es aquella en la cual la impresion es a doble espacio.

Ejemplo:

```
        #
[1]---> WRITE EJEMPLO/WRITE:DOUBLE
        #RUNNING 5947
        #
```

[1] : En el ejemplo se observa el uso de la segunda opcion para imprimir un archivo a doble espacio. El archivo a imprimir es el mismo que el de la opcion uno.

Aqui se muestra la parte del listado en donde se imprimio el archivo.

EJEMPLO/WRITE (04/16/82)

```
100 EJEMPLIFICACION DEL COMANDO WRITE
200 USADO PARA LA IMPRESION DE ARCHIVOS
300 LOS ARCHIVOS NO SON COMPILADOS
400
500 LAS OPCIONES ILUSTRARAN SU USO
600 @@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

A continuacion se presenta el texto que sera manipulado a lo largo de la parte correspondiente a los comandos de actualizacion:

(FIX,DELETE,REPLACE,MOVE,FIND,...)

Este texto como se observara tiene muchos errores, pues con el se ejemplificara el uso de dichos comandos, es decir, se haran todas las modificaciones necesarias para que al final de esta parte se pueda ver que, utilizando dichos comandos, el texto quedo en forma correcta.

Texto erroneo:

```
100.....................................................
200!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
300.
400.....................................................
500#####################################################
600          EL PERO I SU AMIGO
700 SOBRE EL ANGOSTO PUENTE DE UN RIO
800 PASABA UN POBRE PERRO,
900 LLEBANDO UNA HUESO EN EL*HOCICO.
1000 MIRANDO HASIA ABAJO, VIO SU PROPIA AMIGO
1100 REFLEJADA EN EL AGUA NEGRA
1200 Y DIJO PARA SI:
1300            "HOLA"
1400             -!!!
1500            "HOLA"
1600
1700 !! CUANTO MEJOR QUE EL MIO DEBIERA
1800 SER EL UESO QUE YEVA ESE
1900 FORTUNADO COMPANERO !!
2000 Y CAYO AL FONDO DEL RIO, QUEDANDOSE
2100 EL POBRE PERRO SIN NINGUNO
2200 Y TRATO DE ARREVATARSELO.
2300 MAS AL ABRIR L$ VOCA, EL HUESO
2400 ESCAPO DE ENTRE SUS DIENTES
```


En esta hoja se presenta el texto en forma correcta, es decir, el texto tal cual debe quedar una vez hechas las modificaciones pertinentes mediante el uso de los comandos de actualizacion.

Estas modificaciones seran las que se utilizaran para ejemplificar su estructura y su funcion.

Texto corregido:

```
100          EL PERRO Y SU SOMBRA
200 SOBRE EL ANGOSTO PUENTE DE UN RIO
300 PASABA UN POBRE PERRO,
400 LLEVANDO UN HUESO EN EL HOCICO
500 MIRANDO HACIA ABAJO, VIO SU PROPIA SOMBRA
600 REFLEJADA EN EL AGUA CLARA
700 Y DIJO PARA SI:
800
900 -! CUANTO MEJOR QUE EL MIO DEBE
1000 SER EL HUESO QUE LLEVA ESE
1100 AFORTUNADO COMPANERO !-
1200 Y TRATO DE ARREBATARSELO.
1300 MAS AL ABRIR LA BOCA, EL HUESO
1400 ESCAPO DE ENTRE SUS DIENTES
1500 Y CAYO AL FONDO DEL RIO, QUEDANDOSE
1600 EL POBRE PERRO SIN NINGUNO.
```

NOTACION:

Esta notacion sera la que se utilizara para todos los comandos de actualizacion.

<Del>: Es un delimitador (ver al final de esta parte la lista de caracteres que pueden ser usados como delimitadores)

<Texto Original>: Conjunto de caracteres que se desea cambiar.

<Texto Nuevo>: Conjunto de caracteres nuevos que van a sustituir al texto original.

<Col 1>: Columna inicial.

<Col 2>: Columna final.

NOTA:

En muchos comandos se puede hacer uso de la opcion de salida "T". "T" indica que se liste(n) la(s) linea(s) donde se cumplio lo que indica el comando.

A continuacion se proporciona una lista de los comandos de actualizacion mas utilizados.

FIX (5 opciones)

FIND (4 opciones)

REPLACE (5 opciones)

MOVE (2 opciones)

INSERT (4 opciones)

DELETE (3 opciones)

RESET (3 opciones)

UPDATE (1 opcion)

De acuerdo a lo anterior se tienen un total de 27 opciones de comandos para actualizar los archivos.

FIX

El comando FIX es utilizado en aquellos casos en que es necesario efectuar modificaciones en determinados registros (lineas) del archivo.

OPCIONES:

DENTRO DE SECUENCIA.

1) * <Secuencia> <Del> <Texto Original> <Del> <Texto Nuevo>

2) * =<Del> <Texto Original> <Del> <Texto Nuevo>

FUERA DE SECUENCIA.

3) FIX <Secuencia> <Del> <Texto Original> <Del> <Texto Nuevo>

4) FIX =<Del> <Texto Original> <Del> <Texto Nuevo>

5) FIX <Secuencia> <Col 1> - <Col 2> <Del> <Texto Original> <Del>

<Texto Nuevo>

Todas estas opciones en su estructura tienen un patron en comun, el cual significa lo siguiente:

En el registro (linea) con numero de secuencia indicada donde dice "texto original", que diga "texto nuevo".

Cuando se utiliza el comando se puede usar la palabra completa FIX o una F. Cualquiera de las dos anteriores puede ser substituida por un asterisco (*).

FIX = F = *

DENTRO DE SECUENCIA

OPCION 1:

.* <Secuencia> <Del> <Texto Original> <Del> <Texto Nuevo>

El FIX dentro de secuencia significa que estando en secuencia automatica (ver comando SEQ) se pueden hacer modificaciones a registros anteriores o al penultimo registro sin salirnos de la misma.

Si la modificacion al corregir los errores no se hace con secuencia automatica, sera necesario salirse de ella con un doble RETURN, hacer las correcciones y nuevamente entrar en secuencia, teniendo perdida de tiempo.

Como se ve esta forma del comando FIX es muy util al estar creando un nuevo archivo.

Se debe tener cuidado con el comando FIX al utilizarlo dentro de secuencia ya que unicamente se puede realizar un FIX de las formas indicadas en los ejemplos que se daran mas adelante. Es conveniente indicar que si la estructura del comando no es correcta no se obtendra error alguno y sera tomado como parte del texto.

Ejemplo 1:

FIX 100/HOLA/ADIOS

Que indica:

En la linea 100 donde dice "HOLA" que diga "ADIOS".

Ejemplo 2:

```
        SE0 600
        600 EL PERO I SU AMIGO
        700 SOBRE EL $NGOSTO PUENTE DE UN RIO
[1]---> 800 %600/PERO/PERRO
[2]---> #
[3]---> 800 %600/I/Y
        #
        800 PASABA UN POBRE PERRO
        900
        #
        LIST
        600 EL PERRO Y SU AMIGO
        700 SOBRE EL $NGOSTO PUENTE DE UN RIO
        800 PASABA UN POBRE PERRO
        #
```

[1] : Aqui se esta creando un archivo y cuando se va a escribir la linea 800 se descubre que hay un error en la linea 600, por lo cual se efectua un FIX a dicha linea (en la misma linea 800).

[2] : Posteriormente aparece el simbolo (#) que es indicacion de que ya ha sido realizado el cambio e inmediatamente despues aparece el numero de secuencia nuevamente, es decir, la linea que corresponda editar (800) y la que se utilizo previamente para efectuar el FIX vuelve a aparecer (una vez que este comando ya se haya ejecutado) teniendo disponible la linea para ser llenada con el texto correspondiente al mismo originalmente antes de empezar la correccion, el FIX solo fue un contenido temporal.

[3] : Despues de hacer lo anterior se descubre que la linea 600 tiene otra parte equivocada pues se puso "I" en lugar de "Y", por lo cual al volverse a poner el numero de secuencia 800 se vuelve a ejecutar otro FIX a la linea 600.

Si al construir una estructura del FIX (debido a que en el texto original hubo equivocaciones) se trata de cambiar un texto, caracter o palabra que no se encuentre dentro del registro, la computadora no manda mensaje de error al ejecutar el comando, esto se observa al tratar de listar la linea con un "L=", ya que solo manda un "#" se haya o no realizado con exito el FIX.

Ejemplo 3:

```
[1]---> 1000 MIRANDO HASTA AVAJO, VIO SU PROPIA AMIGO
[2]---> 1100 %1000/AVAJO/ABAJO
[3]---> #
[4]---> 1100
        #
[5]---> L=
        #NO TARGET TO FIX %1000:"AVAJO"
        1000 MIRANDO HASTA AV$JO, VIO SU PROPIA AMIGO
        #
```

[1] : Simboliza la parte de las lineas anteriores, pues se supone que se esta editando (en secuencia automatica).

[2] : Aunque la estructura del FIX esta correcta, no funcionara puesto que en la linea 1000 no hay ninguna palabra "AVAJO".

[3] : Pero la respuesta que se manda es como en los casos anteriores.

[4] : Se abandona la secuencia automatica para verificar.

[5] : Al tratar de listar la linea que se intento modificar con el comando "L=", la terminal manda un mensaje "No hay texto 'AVAJO' en esta linea" y lista la original sin corregir. Es decir, cuando no encuentra la palabra indicada la terminal contesta unicamente con un "#" y para saber si efectuo el cambio es necesario listar la linea.

En el siguiente ejemplo se ilustra la falta de un delimitador y que la computadora no manda error, sino que el FIX fue lo-

sado como contenido del registro, se puede comprobar esto al observar que el numero de secuencia es progresivo.

Ejemplo 4:

```
L
600 EL PERRO Y SU AMIGO
700 SOBRE EL $NGOSTO PUENTE DE UN RIO
800 PASABA UN POBRE PERRO
900 LLEBANDO UNA HUESO EN EL HOCICO
#
S 1000
1000 $700/$NGOSTO ANGOSTO
1100 MIRANDO HASIA AV$JO, VIO SU PROPIA AMIGO
1200
#
```

OPCION 2:

.$ = <Del> <Texto Original> <Del> <Texto Nuevo>

El FIX= unicamente sera efectuado en aquellos casos en que se quiera modificar el penultimo registro (el registro anterior a la secuencia aparecida), tomando en consideracion que la estructura del FIX sea correcta, de lo contrario sera tomada como parte del registro.

Ejemplo:

```
800 PASABA UN POBRE PERRO
900 LLEBANDO UNA UESO EN EL HOCICO
1000 $=/LLEBANDO/LLEVANDO
#
1000 $=/UESO/HUESO
#
1000
#
L 900
900 LLEVANDO UNA HUESO EN EL HOCICO
#
```

FUERA DE SECUENCIA.

La accion es identica al comando utilizado dentro de secuencia. Se utiliza cuando ya se tiene el archivo completo y en algunos registros se quiere modificar el texto.

OPCION 3:

FIX <Secuencia> <Del> <Texto Original> <Del> <Texto Nuevo>

Ejemplo 1:

```
L 1000
1000 MIRANDO HASIA AV$JO, VIO SU PROPIA AMIGO
#
FIX 1000/HASIA/HACIA
#
L=
1000 MIRANDO HACIA AV$JO, VIO SU PROPIA AMIGO
#
```

A continuación se dan tres ejemplos en los cuales se muestra la estructura del FIX y se podrá observar que los cambios son realizados en forma correcta en los dos ultimos casos.

Ejemplo con error:

```
[1]---> L 1100
        1100 REFLEGADO EN EL AGUA NEGRA
        #
[2]---> FIX 1100/REFLEGADO REFLEJADA
        #NO MATCHING DELIMITER
```


Ejemplos correctos:

```
[1]---> L 1100
        1100 REFLEGADO EN EL AGUA NEGRA
        #
[2]---> FIX 1100/REFLEGADO/REFLEJADA
        #
[3]---> L=
        1100 REFLEJADA EN EL AGUA NEGRA
        #
```

```
[1]---> L 2100
        2100 EL POBRE PERRO SIN NINGUNO
        #
[2]---> $2100/NINGUNO/NINGUNO
        #
[3]---> L=
        2100 EL POBRE PERRO SIN NINGUNO
        #
```

[1] : Primero se lista la linea donde se va realizar el cambio de texto.

[2] : La estructura del FIX requerida, en el primer ejemplo la estructura esta incorrecta (falta un delimitador).

[3] : Usando el comando "L=" (ver comando LIST) se verifica que el cambio haya sido realizado correctamente.

CION 4:

FIX=<Del> <Texto Original> <Del> <Texto Nuevo>.

Esta estructura se utiliza cuando en una linea se quieren corregir mas de dos errores diferentes.

Ejemplo:

```
        L 1500
        1500 SER EL UESO QUE YEVA ESE
        #
[1]---> FIX 1500/UESO/HUESO
        #
[2]---> FIX=/YEVA/LLEVA
        #
        L=
        1500 SER EL HUESO QUE LLEVA ESE
        #
```

[1] : El primer error se corregira con la forma del FIX mencionada en el inciso 2.1.

[2] : Como se esta actualizando un registro de nuestro "Workfile" (archivo de trabajo) el apuntador (direccion de la linea del archivo que maneja el editor CANDE) se queda en ese registro, con el signo "=" se hace mencion de que se trata de la linea con secuencia dada previamente con el primer FIX, no habiendo necesidad de poner el numero de secuencia para FIX sucesivos de la misma linea.

OPCION 5:

FIX <Secuencia><Col 1>-<Col 2><Del><Texto Original><Del>
<Texto Nuevo>.

Con esta estructura del FIX se selecciona un intervalo de columnas. En caso de que no se especifique la columna 2 (solo la columna 1) entonces sirve para cambiar un caracter, o en caso de que si se especifique la columna 2, se esta pidiendo que la correccion del texto se deba hacer dentro de ese intervalo de columnas haciendo mas selectivo este cambio.

Ejemplo:

```
L 300
300 . . . . . .
#
FIX 300 5-6/.//&
#
L*
300 . . & . . . .
#
```


FIND

El comando FIND ayuda a localizar caracteres o palabras en un archivo del cual se desconoce la(s) secuencia(s) en donde se encuentra la informacion buscada. Si este comando no existiera, para localizar la informacion se tendria que listar el archivo completo y si este es muy grande, seria un proceso muy laborioso y esto quitaria mucho tiempo.

OPCIONES:

1) FIND <Del><Texto><Del>

2) FIND <Del><Texto><Del>,T

3) FIND <Del> <Texto 1> <Del>,...,<Del> <Texto N> <Del>,T

4) FIND LIT<Del> <Texto> <Del> <Col 1>-<Col 2>,
F<Nombre de Archivo>

OPCION 1:

FIND <Del> <Texto> <Del>

Por medio de esta opcion, se puede buscar un caracter o palabra obteniendose como respuesta el numero de secuencia(s) en donde se encontro la informacion deseada.

Con esta opcion no se muestra el registro con su contenido.

Ejemplo: (Estructura incorrecta, falta un delimitador)

```
FIND/AMIGO
#NO MATCHING DELIMITER
```

Estructura correcta:

```
      FIND/AMIGO/
[1]---) #WORKFILE ARTURO/TEXTO
       600, 1000
       #
```

[1] : Aqui contesta con los numeros de las lineas donde encontro la palabra "AMIGO".

En esta forma se obtienen los numeros de los registros y se tiene que usar el comando LIST para ver su contenido, empleando instrucciones de mas. Con la instruccion que se da a continuacion se evita lo anterior.

OPCION 2:

FIND <Del> <Texto> <Del> : T

Esta opcion tiene la misma funcion que el inciso 1, con la diferencia de que buscara y listara los registros en donde se encuentra la informacion requerida.

Ejemplo: (Estructura incorrecta, falta la opcion de salida).

```
FIND/AMIGO/:
#OUTPUT OPTION EXPECTED
```

Estructura correcta:

```
FIND/AMIGO/:T
#WORKFILE ARTURO/TEXTO
600 EL PERRO Y SU AMIGO
1000 MIRANDO HACIA ABAJO, VIO SU PROPIA AMIGO
#
```

La opcion de salida para este comando tiene dos formas: la primera es el " :T " y la segunda " :F (nom de archivo).

NOTA: VER OPCION # 4 DEL COMANDO FIND

OPCION 3:

FIND<Del> <Texto 1> <Del>,...,<Del> <Texto N> <Del> :T

Por medio de esta opcion se puede efectuar la busqueda de mas de un caracter o palabra diferente, cada texto debe ponerse entre delimitadores separados por comas, es necesario poner los delimitadores por ser este un medio para separar un texto del

otro, de otra forma nos dara un error como se muestra en el primer ejemplo.

Ejemplo: (Estructura incorrecta, falta un delimitador).

```
FIND/HUESO/,/AGUA/, PERRO/:T
#STRING DELIMITER EXPECTED, SCANNING PERRO
```

Estructura correcta:

```
FIND/HUESO/,/AGUA/,/PERRO/:T
#WORLFILE ARTURO/TEXTO
600 EL PERRO Y SU AMIGO
800 PASEAN UN POBRE PERRO,
900 LLEVANDO UNA HUESO EN EL HOCICO.
1100 REFLEJADA EN EL AGUA NEGRA
1500 SER EL HUESO QUE LLEVA ESE
2100 EL POBRE PERRO SIN NINGUNO
2300 MAS AL ABRIR LA BOCA, EL HUESO
#
```

Como se ve, el orden en que aparecen los registros en donde se encuentra la informacion no aparece en el orden en que se colocaron en la instruccion, sino que se revisa el archivo desde el inicio, los registros aparecen conforme se va encontrando la informacion en los registros revisados.

OPCION 4:

FIND LIT(Del) (Texto) (Del)(Col 1)-(Col 2); F(Nom. de Archivo)

Esta opcion es mas selectiva en la busqueda de caracteres, palabras o textos. Se puede hacer que se busque en una columna o intervalo de columnas. El texto entre los delimitadores se buscara en forma identica dentro del registro o archivo sin tomar en

cuenta si hay o no blanco a continuacion. Si el archivo es muy grande y no se desea errorar a que la terminal liste los registros, con la opcion de salida " F: ", se pueden almacenar en un archivo en disco los registros que contienen la informacion buscada. Para consultar el archivo posteriormente y hacer los cambios necesarios se debe listar dicho archivo.

Ejemplo: (Estructura en la cual falta el LIT).

```
[1]---> FILE NUEVO
        #NO FILE(S) ON PACK
        #
[2]---> FIND/FORTU/21-20:F NUEVO
        #WORLFILE ARTURO/TEXTO
        #
[3]---> FILE NUEVO
        #NO FILE(S) ON PACK
        #
```

Estructura con el LIT:

```
[2]---> FIND LIT/FORTU/21-20:FILE NUEVO
        #WORLFILE ARTURO/TEXTO
        #
[3]---> FILE NUEVO
        (CSCU) ON PACK
        . NUEVO : SEQDATA
        #
[4]---> L NUEVO
        #FILE (CSCU)NUEVO ON PACK
        1900 FORTUNADO COMPANERO IX
        #
```

[1] : Con el comando FILE verifica si existe el archivo NUEVO. Si existe, el comando FIND no es ejecutado, indicandose que el archivo ya existe. En este caso se indica que no existe, por lo tanto el FIND si se puede realizar.

[2] : En la primera estructura sin el LIT, no se encuentra el texto "FORTU" debido a que no existe como palabra aislada, el texto forma parte de la palabra "AFORTUNADO". Con el LIT buscara exactamente el texto "FORTU" tomandola como una palabra existente sin importar los caracteres que se encuentren a continuacion.

[3] : Aqui se verifica si el archivo fue creado. Si es asi se puede consultar posteriormente al necesitarlo.

[4] : Consulta del archivo NUEVO.

En la sintaxis anterior tenemos que:

<Col 1> - <Col 2>:

Si la columna 2 es omitida, la busqueda sera solo de un caracter, ya que solo se trata de una columna. Si se da la columna 2, se buscara en el intervalo dado por la columna inicial y la final.

LIT :

Busca exactamente el texto indicado, sin importar si este esta como palabra completa o solo es parte de una palabra o palabras.

:F <Nom. de Arch.> :

Almacena en un archivo en disco el registro o grupo de registros donde se encontro el texto buscado, conjuntamente con su contenido.

NOTA:

Todas las opciones anteriores del comando FIND asumen que la palabra buscada es una palabra COMPLETA, es decir, va entre blancos y por tanto se considera aislada. Por eso se tiene esta ultima opcion la cual se utiliza para buscar PORCIONES de palabras (es decir, el texto indicado no se considerara aislado, o lo que es lo mismo, no se encontrara necesariamente entre blancos).

MOVE

En ocasiones se observa que determinados registros del archivo no van en el lugar donde se encuentran, sino en secuencias anteriores o posteriores. Para solucionar este problema se recomienda usar el comando MOVE (tambien se puede utilizar su abreviatura MO), que permite mover un registro o un conjunto de registros a cualquier parte del archivo.

Se debe tener precaucion de no moverlas hacia un registro que ya existe, puesto que no se llevara a cabo el movimiento. Asi mismo aunque resulte redundante, es conveniente mencionar que los registros que sean movidos a otra parte del programa, desapareceran de su lugar y se encontrarán en la secuencia hacia donde fueron movidos.

OPCIONES:

1) MOVE <Secuencia> TO <Secuencia N>

2) MOVE <Secuencia 1>-<Secuencia 2> TO <Secuencia 3>+<Incremento>

OPCION 1:

MOVE <Secuencia> TO <Secuencia N>

Esta opcion solo permite mover un solo registro hacia otra parte del archivo. Se observara mas claramente con el siguiente ejemplo.

Ejemplo:

```
[1]---> L300
        300
        #
[2]---> MOVE 300 TO 1000
[3]---> #UPDATING
        #SEQ RANGE OVERLAP
        #MOVE NOT DONE
        #
[4]---> MOVE 300 TO 1050
        #UPDATING
        #
[5]---> L 300
        #
[6]---> L1050
        1050
        #
```

[1] : Se listo la linea numero 300

[2] : se trato de mover el registro numero 300 hacia el registro numero 1000.

[3] : Mensajes que manda la computadora, indicando que el movimiento no pudo ser efectuado, ya que la secuencia hacia donde se intento mover el registro ya existe.

[4] : Se realiza el movimiento hacia otra linea.

[5] : Se listo la linea 300 y se observa que ya no contiene nada.

[6] : Se lista la nueva linea y se observa que tiene el contenido de la linea original, en este caso la linea 300.

NOTA:

OVERLAP. - Significa que hay un traslare, es decir, se trata de mover una linea a otra que ya existe.

OPCION 2:

MOVE (Secuencia 1)-(Secuencia 2) TO (Secuencia 3)+(Incremento)

Con esta opcion se puede mover un conjunto de registros dados desde la secuencia inicial (secuencia 1) hasta la secuencia final (secuencia 2) y quedaran colocadas a partir de la secuencia tres (secuencia 3), que es tomada como base, asi mismo se indica el incremento (o sea, los intervalos de las secuencias). El incremento dado por omision es de 100.

Ejemplo 1:

```
[1]---> L2000-2100
        2000 Y CAYO AL FONDO DEL RIO, QUEDANDOSE
        2100 EL POBRE PERRO SIN NINGUNO.
        #
[2]---> MOVE 2000-2100 TO 2400
        #UPDATING
        #SEQ RANGE OVERLAP
        #MOVE NOT DONE
        #
```

[1] : Se listo el numero de secuencia de los registros que se van a mover.

[2] : Se hace el MOVE de los registros anteriores hacia el numero de secuencia 2400, pero la computadora indica que el movimiento no puede ser efectuado, puesto que el registro 2400 ya existe.

Ejemplo 2:

```
[3]---> MOVE 2000-2100 TO 2410+10
        #UPDATING
        #
[4]---> L2000-2100
        #
[5]---> L2410-2500
        2410 Y CAYO AL FONDO DEL RIO, QUEDANDOSE
        2420 EL POBRE PERRO SIN NINGUNO.
        2500
        #
```

[3] : Se realizo nuevamente el MOVE, ahora hacia una secuencia que no existia. Ya que no se recibio ningun mensaje se tendra la certeza que fue ejecutado.

[4] : Se listo el numero de secuencia de los registros que se movieron, aparece inmediatamente el "#", significando esto que no tiene ningun contenido.

[5] : Se listo el numero de secuencia a partir de donde se movio y se observa que aparece una nueva secuencia y el contenido es el que se tenia originalmente en la secuencia 2000-2100.

INSERT

A traves de este comando se puede insertar lineas del archivo a otra parte del mismo o bien lineas de un archivo a otro con la caracteristica de que las lineas que sean insertadas no desapareceran de su lugar de origen. Se debe tener cuidado de no tratar de insertar lineas en secuencias ya existentes (ya que tienen como contenido un texto), y el comando no sera ejecutado.

OPCIONES:

1) INSERT <Secuencia> AT <Secuencia N>

2) INSERT <Secuencia 1>-<Secuencia 2> AT <Secuencia N>

3) INSERT <Secuencia 1>-END AT <Secuencia N>+<Incremento>

4) INSERT <Secuencia 1>-<Secuencia 2> <Nombre del Archivo>
AT <Secuencia N>

Actualmente el archivo se encuentra de la siguiente manera:

```
L
100 .....................................................
200 -!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
400 .....................................................
500 #####################################################
600             EL PERRO Y SU SOMBRA
700 SOBRE EL ANGOSTO PUENTE DE UN RIO
800 PASABA UNA POBRE PERRO,
900 LLEVANDO UNA  HUESO EN EL HOCICO.
1000 MIRANDO HACIA ABAJO, VIO SU PROPIA AMIGO
1050 .
1100 REFLEJADA EN EL AGUA NEGRA
1200 Y DIJO PARA SI:
1300          "HOLA"
1400           -!!!
1500          "HOLA"
1600
1700 -I CUANTO MEJOR QUE EL MIO DEBIERA
1800 SER EL HUESO QUE LLEVA ESE
1900 AFORTUNADO COMPANERO I-
2200 Y TRATO DE ARREVATARSELO.
2300 MAS AL ABRIR LA BOCA, EL HUESO
2400 ESCAPO DE ENTRE SUS DIENTES
2410 Y CAYO AL FONDO DEL RIO, QUEDANDOSE
2420 EL POBRE PERRO SIN NINGUNO.
2500 .
```

OPCION 1:

INSERT <Secuencia> AT <Secuencia N>

Esta opcion unicamente interpreta la insercion de una sola linea.

Ejemplo:

```
[1]---> L500
        500 ##########################################
        #
[2]---> INSERT 500 AT 2510
        #UPDATING
        #
[3]---> L500
        500 ##########################################
        #
[4]---> L2510
        2510 #########################################
        #
```

[1] : Se lista el registro con numero de secuencia 500, ya que es el que se desea insertar.

[2] : Se realizo el insert de la secuencia 500 en la 2510, indicando que ya fue ejecutado este.

[3] : se lista nuevamente la linea 500 con la finalidad de que se pueda observar que no ha desaparecido

[4] : Se lista la linea 2510, y se observa que tiene el mismo contenido de la linea 500.

OPCION 2:

INSERT <Secuencia 1>-<Secuencia 2> AT <Secuencia N>

En la opcion anterior solo se podia insertar una linea, ahora con esta nueva opcion se podra insertar un conjunto de mas de dos lineas.

Ejemplo:

```
[1]---> L 1300-1500
        1300            "HOLA"
        1400             -###
        1500            "HOLA"
        #
[2]---> INSERT 1300-1500 AT 2600
        #UPDATING
        #
[3]---> L 1300-1500
        1300            "HOLA"
        1400             -###
        1500            "HOLA"
        #
[4]---> L 2600-2800
        2600            "HOLA"
        2700             -###
        2800            "HOLA"
        #
```

[1] : Se lista el conjunto de lineas que se desean insertar.

[2] : Se realiza el insert de un conjunto de registros hacia otra secuencia, y la computadora indicara que ya han sido realizados.

[3] : Se lista el conjunto de registros que inicialmente se insertaron y se observa que aun permanecen con su informacion.

[4] : Se listan los registros insertados y se observa que contienen la misma informacion que los registros anteriores.

OPCION 3:

INSERT <Secuencia>-END AT <Secuencia N> + <Incremento>.

Se puede observar que esta estructura del insert contiene un END y un incremento. El END indica el final del archivo. El incremento indica los intervalos entre las secuencias de los registros.

Ejemplo:

```
[1]---> L 2600-END
        2600          "HOLA"
        2700           -fff
        2800          "HOLA"
        #
[2]---> INSERT 2600-END AT 10+5
        #UPDATING
        #
[3]---> L 2600-END
        2600          "HOLA"
        2700           -fff
        2800          "HOLA"
        #
[4]---> L 10-100
        10          "HOLA"
        15           -fff
        20          "HOLA"
        100 .....
        #
```

[1] : Se listan los registros con numero de secuencia 2600 hasta el final del archivo.

[2] : Se realiza el comando insert. Se insertara del registro 2600 hasta el final en la linea 10 con incrementos de 5.

[3] : Se lista nuevamente el conjunto de registros que se insertaron y se observa que aun conserva su contenido.

[4] : Se lista desde la secuencia 10 hasta la 100 con la finalidad de comprobar la ejecucion de la insercion y se observa que contiene la informacion de los anteriores registros.

OPCION 4:

INSERT <Secuencia 1>-<Secuencia 2> <Nombre del Archivo> AT <Secuencia N>

En algunas ocasiones resulta conveniente insertar registros de un archivo a otro, de ahi que se este haciendo ahora mencion a un nombre de archivo (del cual se tomaran las lineas para insertarlas en el archivo de trabajo). El ejemplo se indica en la siguiente hoja.

emplo:

```
 ]---> L PRUEBA/II
 ]---> #FILE (BUSI)PRUEBA/II ON UNAM1
       100 PROGRAMA DE PRUEBA
       200 OBJETIVO: PRACTICAR LOS COMANDOS DE ACTUALIZACION
       300 PROGRAMA DE FORMACION DE RECURSOS HUMANOS
       400           P. U. C.
       #
 ]---> INSERT 100-400 PRUEBA/II AT 2900
       #UPDATING
       #
 ]---> L
 ]---> 10          "HOLA"
       15           -!!!
       20          "HOLA"
       100 ..................................................
       200 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
       400 ..................................................
       500 ##################################################
       600          EL PERRO Y SU AMIGO
       700 SOBRE EL ANGOSTO PUENTE DE UN RIO
       800 PASABA UN FLACO PERRO,
       900 LLEVANDO UNA HUESO EN EL HOCICO.
       1000 MIRANDO HACIA ABAJO, VIO SU PROPIA AMIGO
       1050
       1100 REFLEJADO EN EL AGUA NEGRA
       1200 Y DIJO PARA SI:
       1300          "HOLA"
       1400           -!!!
       1500          "HOLA"
       1600
       1700 -I CUANTO MEJOR QUE EL RIO DEBIERA
       1800 SER EL HUESO QUE LLEVA ESE
       1900 AFORTUNADO  COMPANERO I-
       2200 Y TRATO DE ARREBATARSELO
       2300 MAS AL ABRIR LA BOCA, EL HUESO
       2400 ESCAPO DE ENTRE SUS DIENTES
       2410 Y CAYO AL FONDO DEL RIO, QUEDANDOSE
       2420 EL POBRE PERRO SIN NINGUNO.
       2500
 ]---> 2510 ##################################################
 ]---> 2600          "HOLA"
       2700           -!!!
       2800          "HOLA"
 ]---> 2900 PROGRAMA DE PRUEBA
       3000 OBJETIVO: PRACTICAR LOS COMANDOS DE ACTUALIZACION
       3100 PROGRAMA DE FORMACION DE RECURSOS HUMANOS
       3200           P. U. C.
       #
```


[1] : Se lista un conjunto de registros del archivo PRUEBA/II que seran tomados para insertarlos en otro.

[2] : La maquina responde: es un archivo que esta bajo la clave (BMBI) y el nombre del archivo el cual esta almacenado en UNAM1.

[3] : Se realiza el INSERT del conjunto de registros del archivo PRUEBA/II a la secuencia 2900 del archivo con el que se esta trabajando.

[4] : Se lista todo el archivo para observar y al mismo tiempo comprobar que los registros fueron insertados a partir del registro con numero de secuencia 2900.

En el ejemplo se puede observar que si se realizaron los Insert de las opciones anteriores:

[A] ----------EJEMPLO OPCION 1

[B] ----------EJEMPLO OPCION 2

[C] ----------EJEMPLO OPCION 3

[D] ----------EJEMPLO OPCION 4

REPLACE

En algunas ocasiones se va a necesitar cambiar un mismo caracter, palabra o texto en varios registros del archivo o en uno solo, para esto es cuando resulta muy util el uso del comando REPLACE. De no haber este comando nos veriamos obligados a listar el archivo completo e ir corrigiendo registro por registro con el comando FIX, efectuando los cambios requeridos, procedimiento que resultaria demasiado laborioso.

OPCIONES:

1) REPLACE <Del><Texto Original><Del><Del><Texto Nuevo><Del>:T

2) REPLACE <Del><Texto Original><Del><Del><Texto Nuevo><Del>
<COL 1>-<COL 2>:T

3) REPLACE <Del><Texto Original 1><Del><Del><Texto Nuevo 1><Del>,
<Del><Texto Original 2><Del><Del><Texto Nuevo 2><Del>
<COL 1>-<COL 2>:T

4) REPLACE <Del><Texto Original><Del><Del><Texto Nuevo><Del>
<COL 1>-<COL 2><Secuencia 1>-<Secuencia 2>:T

5) REPLACE F LIT <Del><Texto Original><Del><Del><Texto Nuevo><Del>
<COL 1>-<COL 2>:F <Nombre del Archivo>

En las orciones se observa que todas ellas tienen ":T", parte de la estructura que puede ir o no, esta orcion listara las lineas en donde se realizaron los cambios, al no ponerla solo obtendremos un "#" y posteriormente se debera listar el archivo en caso de que se deseen verificar los cambios, si tenemos la certeza de que los cambios se le indicaron en forma correcta, no es necesario utilizar la orcion de salida ":T".

El patron comun en todas las orciones significa: reemplaza todo donde diga <Texto Original> por <Texto Nuevo>, es decir es en si un FIX de cierto texto para todo el archivo.

OPCION 1:

REPLACE <Del><Texto Original><Del><Del><Texto Nuevo><Del>:T

Esta orcion permite cambiar un caracter o palabra por otro nuevo que le indiquemos, el ":T" indica que despues de que efectuo el reemplazo nos muestre el registro o los registros donde lo realizo. Como se puede observar los delimitadores son cuatro, se debera usar el mismo delimitador para toda la estructura, cada uno de los textos (el original y el nuevo) debe estar entre delimitadores:

El reemplazo se hara en todos y cada uno de los registros del archivo donde se encuentre el <Texto Original> (Palabra o caracter)

Ejemplo:

Estructura incorrecta, falta un delimitador:

```
        REP/AMIGO/SOMBRA/*T
[1]---> #STRING DELIMITER EXPECTED, SCANNING SOMBRA
```

[1] : La maquina nos indica que no pudo efectuar el reemplazo debido a que hay un error, el cual es la omision de un delimitador en la estructura del comando.

Estructura correcta:

```
REP/AMIGO/SOMBRA/*T
#MODIFILE ARTURO/TEXTO
#UPDATING
600 EL PERRO Y SU SOMBRA
1000 MIRANDO HACIA ABAJO, VIO SU PROPIA SOMBRA
#
```

Como se ve en el ultimo ejemplo se actualiza el archivo y a la vez se van obteniendo los registros en donde se realizaron los cambios con "*T".

OPCION 2:

2.1) REPLACE <Del><Texto Original><Del><Del><Texto Nuevo><Del>
<Col>*T

Esta opcion es mas selectiva debido a que se va a hacer el reemplazo solo en la columna especificada para todos los registros del archivo.

2.2) REPLACE<Del><Texto Original><Del><Del><Texto Nuevo><Del>
<Col 1>-<Col 2>*T

Esta opcion tambien es selectiva pues permite efectuar el reemplazo en un intervalo de columnas para todos y cada uno de los registros del archivo donde encuentre el <Texto Original> indicado.

En ambas opciones el listado de los registros modificados solo se obtendra en caso de haber utilizado la opcion "*T".

Ejemplo:

Estructura invalida, falta un delimitador:

```
REP/UNA/UN/210-15*T
#STRING DELIMITER EXPECTED, SCANNING UN
```

Estructura correcta:

```
REP/UNA//UN /210-15*T
#XORIFILE ARTURO/TEXTO
#UPDATING
900 LLEVANDO UN HUESO EN EL HOCICO.
#
```

OPCION 3:

REPLACE <Del><Texto Original 1><Del><Del><Texto Nuevo 1><Del>,
<Del><Texto Original 2><Del><Del><Texto Nuevo 2><Del>
<COL 1>-<COL 2>:T

Como se puede observar esta estructura del REPLACE contiene comas (",") lo que indica que se pueden realizar varios reemplazos en diferentes registros mediante una sola instruccion. la estructura indica lo siguiente:

Reemplaza <Texto Original 1> por <Texto Nuevo 1> y <Texto Original 2> por <Texto Nuevo 2>. De la columna 1 (inicial) a la columna 2 (final) e indicara las secuencias donde fueron realizados los cambios con el contenido de las mismas.

Ejemplo:

Estructura invalida, falta un delimitador:

```
REP/NEGRA//CLARA/, /DEBIERA/DEBE/@30-50:T
#STRING DELIMITER EXPECTED. SCANNING DEBE
```

Estructura correcta:

```
REP/NEGRA//CLARA/, /DEBIERA//DEBE/@20-50:T
#WORKFILE ARTURO/TEXTO
#UPDATING
1100 REFLEJADA EN EL AGUA CLARA
1700 EL CUANTO MEJOR QUE EL MIO DEBE
#
```


Como se explico anteriormente en la opcion 2, si la Columna 2> (columna final) no es especificada, el reemplazo sera efectuado solamente sobre el caracter indicado.

OPCION 4:

REPLACE F LIT <Del><Texto Original><Del><Del><Texto Nuevo><Del>
<COL 1>-<COL 2><Secuencia 1>-<Secuencia 2> :T

El proceso de reemplazo con esta opcion va a ser en forma mas selectiva. la F es una abreviatura de FIRST que indica el primer caracter, palabra o texto que contenga al texto original que se encuentre. El "LIT" busca en el registro porciones de texto o palabras no importando que no se encuentren aisladas, es decir que no se formen palabras completas, solamente porciones de estas. Para las columnas se cuenta con secuencia 1 y secuencia 2 las cuales indican el intervalo de registros donde se desea trabajar, en caso de que no se especifique la secuencia 2, el reemplazo se hara solamente en linea indicada por secuencia 1.

El F LIT nos hace el reemplazo del <Texto Original> pero del primero que se encuentre en el registro. A diferencia de las opciones anteriores aqui hara reemplazo si hay mas de un <Texto Original> repetido en el mismo registro, pues solo lo hara en la primera ocurrencia que encuentre en la linea.

Ejemplo:

Estructura sin el F LIT:

```
C13---> REP/00//60/@15-30 2300:T
        #WORKFILE ARTURO/TEXTO
        #UPDATING
        #
```

Estructura usando el F LIT:

```
E23---> REP F LIT/UO//BO/@15-30 2300:T
        #WORKFILE ARTURO/TEXTO
        #UPDATING
        2300 MAS AL ABRIR L# BOCA, EL HUESO
        #
```


E13 : En el texto original, en la linea 2300 del archivo no existe la palabra "UO" por lo cual en el primer ejemplo no hace el reemplazo, la palabra que quiere modificar es "UOCA".

E23 : En esta estructura con el F LIT si se hace la modificacion, ya que no le importa si "UO" es palabra completa (que es el caso cuando no se usa el F LIT) o solo parte de una palabra (que es el caso y utilidad del F LIT).

En los ejemplos anteriores omitimos la secuencia 2 por lo que el reemplazo se llevo a cabo solamente en el registro indicado.

En el siguiente ejemplo se hara el reemplazo en un intervalo de registros.

Ejemplo 1:

```
L100-500
100 ..............................................
200 ##############################################
400 ..............................................
500 ##############################################
#
REP F LIT/.//%/@30-40 100-400:T
#WORKFILE ARCHIVO
#UPDATING
100 ..............................%...............
400 ..............................%...............
#
```

Ejemplo 2:

```
L1700-1900
1700 ¥I CUANTO MEJOR QUE EL MIO DEBE
1800 SER EL UESO QUE YEVA ESE
1900 FORTUNADO CONF¥NERO I¥
#
E13---> REP F LIT/I//¥/1700-1900:T
#WORKFILE ARTURO/TEXTO
#UPDATING
1700 ¥I CUANTO MEJOR QUE EL MIO DEBE
1900 FORTUNADO CONF$NERO I¥
#
```

NOTA:

En el ejemplo 2 se observa que no es necesario indicar el intervalo de la columna, ya que el usuario sabe para este caso que donde desea realizar el cambio es en la primera ocurrencia del registro 1700 y del 1900.

OPCION 5:

REPLACE F LIT <Del><Texto Original><Del><Del><Texto Nuevo><Del>
<COL 1>- <COL 2>: F <Nombre del Archivo>

Esta estructura es identica a la opcion 2 en cuanto a su funcionamiento, a excepcion de que en lugar del ":T" ahora se indica ":F <Nombre del Archivo>". Esta opcion almacenara un archivo en disco que contiene los cambios realizados, que posteriormente se podra consultar para verificar que los cambios hayan sido realizados.

Ejemplo 1:

```
[1]---> FILE NUEVO1
        #NO FILE(S) ON PACK
        #
[2]---> REP F LIT/t//-/21-40:F NUEVO1
        #WORKFILE ARTURO/TEXTO
        #UPDATING
        #
[3]---> FILE NUEVO1
        (CSCU) ON PACK
        , NUEVO1 : SERDATA
        #
[4]---> L NUEVO1
        #FILE (CSCU)NUEVO ON PACK
        200 -*********************************************
        1400          --**
        #
```


[1] : Con el comando file se verifica si el archivo existe, esto es con el fin de que muestre que el archivo va a ser creado.

[2] : Se presenta la estructura del REPLACE, con el F LIT es posible observar que en el registro 200 y 1500 existen mas de un caracter del <Texto Original>, pero solo hizo el reemplazo en el primer caracter aunque se le haya indicado el intervalo de columnas de la 1 a la 40.

[3] : A traves del comando file se verifica que el archivo ya esta creado.

[4] : Se listo el archivo ya creado el cual contiene las lineas donde fue realizado el reemplazo.

En el siguiente ejemplo se utiliza la opcion mediante la cual es almacenada la informacion en un archivo con excepcion de que no se le indico la Columna 1 ni la Columna 2. el reemplazo sera realizado en todo el archivo ya que no se le indico la secuencia o el intervalo de registros.

Ejemplo 2:

```
[1]---> FILE NUEV02
        #NO FILE(S) ON PACK
        #
[2]---> REP F LIT/$//A/ F NUEV02
        #UOPEN FILE ARTURO/TEXTO
        #UPDATING
        #
[3]---> FILE NUEV02
        (CSCU) ON PACK
          NUEV02    SEQDATA
        #
[4]---> L NUEV02
        #FILE (CSCU)NUEV02 ON PACK
        700 SOBRE EL ANGOSTO PUENTE DE UN RIO
        1000 MIRANDO HACIA ABAJO, VIO SU PROPIA SOMBRA
        2300 MAS AL ABRIR LA BOCA, EL HUESO
        #
```

[1] : se verifica con el comando file que el archivo NUEV02 no se encuentra en disco, de lo contrario el comando REPLACE no se ejecutara

[2] : con el F LIT solo la primera ocurrencia del Texto Original sera reemplazado, en este caso el caracter "$" por la letra "A".

[3] : se verifica que el archivo fue creado.

[4] : Se lista el archivo para verificar que los cambios hayan sido realizados.

DELETE

En ocasiones resulta necesario borrar o suprimir uno o varios registros que se encuentran dentro del archivo de trabajo, esto es posible realizarlo a traves del comando DELETE o bien mediante su abreviatura DEL.

OPCIONES:

1) DELETE (Secuencia)

2) DELETE (Secuencia 1)-(Secuencia 2)

3) DELETE ALL

Actualmente, debido al contenido que se encuentra en el archivo, es necesario ejecutar aun mas correcciones, las cuales implican borrar una serie de registros para lo cual se procedera a emplear el comando DELETE.

El texto se encuentra de la siguiente manera:

Ejemplo:

```
10              "HOLA"
15               --11
20              "HOLA"
100 .....................................................
200 -!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
400 .....................................................
500 ##################################################
600          EL PERRO Y SU SOMBRA
700 SOBRE EL ANGOSTO PUENTE DE UN RIO
800 PASABA UN POBRE PERRO,
900 LLEVANDO UN HUESO EN EL HOCICO.
1000 MIRANDO  HACIA ABAJO, VIO SU PROPIA SOMBRA
1050
1100 REFLEJADA EN EL AGUA CLARA
1200 Y DIJO PARA SI:
1300            "HOLA"
1400             --11
1500            "HOLA"
1600
1700 -! CUANTO MEJOR QUE EL MIO DEBE
1800 SER EL HUESO QUE LLEVA ESE
1900 AFORTUNADO COMPANERO !-
2200 Y TRATO DE ARREBATARSELO
2300 MAS AL ABRIR LA BOCA, EL HUESO
2400 ESCAPO DE ENTRE SUS DIENTES
2410 Y CAYO AL FONDO DELL RIO, QUEDANDOSE
2420 EL POBRE PERRO SIN NINGUNO.
2500
2510 ##################################################
2600            "HOLA"
2700             -111
2800            "HOLA"
2900 PROGRAMA DE PRUEBA
3000 OBJETIVO:  PRACTICAR LOS COMANDOS DE ACTUALIZACION
3100 PROGRAMA DE FORMACION DE RECURSOS HUMANOS
3200             P. U. C.
```


OPCION 1:

DELETE <Secuencia>

Como se menciono anteriormente este comando permite borrar registros de un archivo. Esta opcion unicamente permite borrar un registro.

Ejemplos: Para el archivo actual necesitamos suprimir las siguientes lineas.

```
[1]---> L100
        100 .....................................
        #
[2]---> DEL100
        #
[3]---> L100
        #
[1]---> L1050
        1050 ....................................
        #
[2]---> DEL1050
        #
[3]---> L1050
        #
[1]---> L2500
        2500 ....................................
        #
[2]---> DEL2500
        #
[3]---> L2500
        #
```

os tres ejemplos anteriores tienen el mismo funcionamiento:

1] : Se listaron los registros que se desean borrar

2] : al aparecer el "#" es indicacion de que ya fue ejecutado el comando DELETE para borrar el registro.

3] : Nuevamente se listo el mismo registro y se obtiene como respuesta un "#", senal de que el registro ya no existe.

tros Ejemplos:

```
      #
1]---> DEL10
       #
       DEL15
       #
       DEL20
       #
       DEL200
       #
       DEL400
       #
       L10
       #
       L15
       #
       L20
       #
       L200
       #
       L400
       #
```

1] : Como se puede observar es sumamente laborioso borrar linea por linea, de ahi que se presentan otras opciones que facilitan este procedimiento.

En el siguiente ejemplo se ilustra otra forma de borrar una linea.

```
[1]---> L500
        500 ##########################################
        #
[2]---> 500
        #
[3]---> L500
        #
```

[1] : Se lista la linea que se desea borrar.

[2] : Indicacion del numero de secuencia y se da un RETURN o un NEW LINE.

[3] : Al tratar de listar la secuencia se observa que ya no existe.

OPCION 2:

DELETE <Secuencia 1>-<Secuencia 2> ///

Con esta nueva estructura del DELETE se estara en posibilidades de borrar un conjunto de registros.

Ejemplo:

```
[1]---> L200-500
        200 ##########################################
        400 ..........................................
        500 ##########################################
        #
[2]---> DEL200-500
        #
[3]---> L200-500
        #
```

[1] : Se lista un conjunto de registros que se desea borrar.

[2] : se borra el conjunto de registros.

[3] : Ahora se listo el conjunto de registros, observando que ya no existen.

El texto queda de la siguiente manera:

```
L
600 EL PERRO Y SU SOMBRA
700 SOBRE EL ANGOSTO PUENTE DE UN RIO
800 PASABA UN POBRE PERRO,
900 LLEVANDO UN HUESO EN EL HOCICO.
1000 MIRANDO HACIA ABAJO, VIO SU PROPIA SOMBRA
1100 REFLEJADA EN EL AGUA CLARA
1200 Y DIJO PARA SI:
1600
1700 -¡ CUANTO MEJOR QUE EL RIO DEBE
1800 SER EL HUESO QUE LLEVA ESE
1900 AFORTUNADO COMPAÑERO !-
2200 Y TRATO DE ARREVATARSELO
2300 MAS AL ABRIR LA BOCA, EL HUESO
2400 ESCAPO DE ENTRE SUS DIENTES
2410 Y CAYO AL FONDO DEL RIO, QUEDANDOSE
2420 EL POBRE PERRO SIN NINGUNO.
#
```


OPCION 3:

DELETE ALL

Por medio de esta opcion se estara indicando que se borren todos los registros del archivo.

Ejemplo:

```
[1]---> DEL ALL
        #
[2]---> L
        #
```

[1] : El comando significa que borre todo.

[2] : Se lista el archivo y se comprueba que todos los registros fueron borrados.

RESEQ

Por medio de este comando se puede resecuenciar el archivo de trabajo.

Este comando es de gran utilidad, puesto que al estar efectuando una serie de modificaciones, se pierde la secuencia original ya que se pudieron haber introducido nuevas lineas (SEQ), efectuar movimientos de lineas (MOVE) o inserciones (INSERT) que la alteraron.)

Se puede resecuenciar el archivo de trabajo escribiendo simplemente RESEQ, aunque tambien es admisible su abreviacion RES, en este caso, la computadora asignara una secuencia con intervalos de 100 en 100, esto significa que la primera linea sera 100, la segunda 200 y asi sucesivamente.

OPCIONES:

1) RESEQ

2) RESEQ <Secuencia Inicial>

3) RESEQ <Secuencia Inicial> , <Incremento>

OPCION 1:

RESEQ

En el siguiente ejemplo se muestra el uso del comando RESEQ.

Secuencia original:

```
LIST
600 EL PERRO Y SU SOMBRA
700 SOBRE EL ANGOSTO PUENTE DE UN RIO
800 PASABA UN POBRE PERRO,
900 LLEVANDO UN HUESO EN EL HOCICO.
1000 MIRANDO HACIA ABAJO, VIO SU PROPIA SOMBRA
1100 REFLEJADA EN AGUA CLARA
1200 Y DIJO PARA SI:
1300
1700 -¡ CUANTO MEJOR QUE EL MIO DEBE
1800 SER EL HUESO QUE LLEVA ESE
1900 AFORTUNADO COMPANERO !-
2000 Y TRATO DE ARREBATARSELO.
2300 MAS AL ABRIR LA BOCA, EL HUESO
2400 ESCAPO DE ENTRE SUS DIENTES
2410 Y CAYO AL FONDO DEL RIO, QUEDANDOSE
2420 EL POBRE PERRO SIN NINGUNO.
#
```


Secuencia despues del RESEQ:

```
RESEQ
#UPDATING
#
LIST
100 EL PERRO Y SU SOMBRA
110 SOBRE EL ANGOSTO PUENTE DE UN RIO
120 PASABA UN POBRE PERRO,
130 LLEVANDO UN HUESO EN EL HOCICO.
140 MIRANDO HACIA ABAJO, VIO SU PROPIA SOMBRA
150 REFLEJADA EN AGUA CLARA
160 Y DIJO PARA SI:
170
180 -¡ CUANTO MEJOR QUE EL MIO DEBE
190 SER EL HUESO QUE LLEVA ESE
200 AFORTUNADO COMPANERO !-
210 Y TRATO DE ARREBATARSELO.
220 MAS AL ABRIR LA BOCA, EL HUESO
230 ESCAPO DE ENTRE SUS DIENTES
240 Y CAYO AL FONDO DEL RIO, QUEDANDOSE
250 EL POBRE PERRO SIN NINGUNO.
#
```

NOTA:

Se hace notar que cuando se le indica un RESEQ, la computadora pondra siempre la secuencia de 100 en 100. Pero si por algun motivo anteriormente se efectuo algun comando en el cual se le haya utilizado algun incremento (por ejemplo 10), este mismo sera tomado como incremento para el comando RESEQ, tal es el caso anterior.

OPCION 2:

RESEQ <Secuencia>

Por medio de esta opcion se puede resecuenciar un archivo, la secuencia inicial que se le indique sera tomada como base y los incrementos por default seran de 100.

En el siguiente ejemplo se puede observar lo anteriormente dicho:

```
[1]---> RESEQ 30
        #UPDATING
        #
[2]---> LIST
[3]---> 30 EL PERRO Y SU SOMBRA
        130 SOBRE EL ANGOSTO PUENTE DE UN RIO
        230 PASABA UN POBRE PERRO,
        330 LLEVANDO UN HUESO EN EL HOCICO
        430 MIRANDO HACIA ABAJO, VIO SU PROPIA SOMBRA
        530 REFLEJADA EN EL AGUA CLARA
        630 Y DIJO PARA SI:
        730
        830 -¡ CUANTO MEJOR QUE EL MIO DEBE
        930 SER EL HUESO QUE LLEVA ESE
        1030 AFORTUNADO COMPANERO !-
        1130 Y TRATO DE ARREBATARSELO.
        1230 MAS AL ABRIR LA BOCA, EL HUESO
        1330 ESCAPO DE ENTRE SUS DIENTES
        1430 Y CAYO AL FONDO DEL RIO, QUEDANDOSE
        1530 EL POBRE PERRO SIN NINGUNO.
```

[1] : Se teclea el comando RESEQ 30.

[2] : Se lista todo el archivo.

[3] : Observese que el 30 es la base y los incrementos son de 100 en 100.

OPCION 3:

RESEQ <Secuencia> + <Incremento>

Ejemplo:

```
[1]---> RESEQ 100+100
        #UPDATING
        #
[2]---> LIST
        100          EL PERRO Y SU SOMBRA
        200 SOBRE EL ANGOSTO PUENTE DE UN RIO
        300 PASABA UN POBRE PERRO,
        400 LLEVANDO UN HUESO EN EL HOCICO.
        500 MIRANDO HACIA ABAJO, VIO SU PROPIA SOMBRA
        600 REFLEJADA EN AGUA CLARA
        700 Y DIJO PARA SI:
        800
        900 -¡ CUANTO MEJOR QUE EL MIO DEBE
        1000 SER EL HUESO QUE LLEVA ESE
        1100 AFORTUNADO COMPANERO !-
        1200 Y TRATO DE ARREBATARSELO.
        1300 MAS AL ABRIR LA BOCA, EL HUESO
        1400 ESCAPO DE ENTRE SUS DIENTES
        1500 Y CAYO AL FONDO DEL RIO, QUEDANDOSE
        1600 EL POBRE PERRO SIN NINGUNO.
        #
```

[1] : Se indica que se de nueva secuencia al archivo tomando como base el 100 y con incrementos de 100 en 100.

[2] : Se lista el archivo para comprobar que los cambios havan sido realizados

UPDATE

Sirve para actualizar archivos.

La actualizacion automatica resulta de los siguientes comandos :

INSERT
MOVE
RESEQ
REPLACE
COMPILE
RUN
TYPE
FIND
SR

La actualizacion de un archivo se da despues de haber ejecutado el comando UPDATE o bien cualquiera de los comandos mencionados en la lista. Por actualizacion se entiende que cada vez que sea ejecutado el comando el archivo que se tenga presente se estara modificando.

Ejemplo:

Indicacion del UPDATE

```
UPDATE
#UPDATING
#
```

Cuando aparece el "#" significa que ya ha sido actualizado el archivo.

Se puede apreciar en los ejemplos de los comandos, que la maquina responde con #UPDATING, esto significa que esta actualizando el archivo, (ver Ejemplos de los comandos mencionados al principio del UPDATE).

* * * DELIMITADORES.

)

(

.

!

+

-

&

$

#

*

'

etc. . .

Como se observa, son todos los caracteres especiales que se encuentran en el teclado de la terminal con excepcion de:

;

,

.

%

arroba(@)

Recuerdese que el delimitador que se use al inicio de cualquier estructura de los comandos de actualizacion debe ser el mismo en el resto de la misma.

COMANDOS PROPIOS DE LA COMPILACION Y EJECUCION

COMPILE

Una vez terminado de editar el archivo de trabajo, si este consiste en un programa ejecutable en algun lenguaje (llamado "programa fuente") y se desea correrlo, sera necesario que la maquina traduzca el lenguaje en el que el programa se hizo, a traves de un compilador (sistema de Software), a un lenguaje de maquina que la computadora comprende, creando para ello un nuevo archivo donde guardara el programa traducido (el cual se llama "programa objeto"). A este proceso se le conoce como "compilacion".

Este proceso se verifica mediante la instruccion COMPILE y es necesario realizarlo para la corrida de cualquier programa. Es por esto que la maquina lo activara automaticamente al darsele la instruccion RUN en el caso de que el programa objeto no fuese aun creado. Por lo tanto, para la ejecucion de una compilacion, contamos con las siguientes opciones:

OPCIONES:

1)COMPILE

2)RUN

OPCION 1:

COMPILE

Instruccion que unicamente realiza la compilacion de un programa. Las opciones de la instruccion son:

1) COMPILE

2) COMPILE WITH <Tipo>

donde el WITH se utiliza en el caso de que se quiera especificar otro compilador distinto al especificado en la instruccion MAKE (por lo cual se implica tambien que es un compilador no propio de la maquina).

En el caso de que sea necesario especificar otro compilador por medio del WITH, sera imposible efectuar la compilacion por medio de la instruccion RUN.

Existe una notable excepcion entre aquellos compiladores no propios de la maquina que es el caso de los programas en lenguaje Pascal, en donde es necesario especificar el compilador.

En el siguiente ejemplo se muestra la manera en que se utiliza la instruccion y la forma en que la maquina responde a ella. Se supone que se tiene presente el archivo de trabajo. (Creado mediante el MAKE o traido del disco mediante el GET).

Ejemplo 1:

```
        #.
[1]---> L
        100 BEGIN
        200 REAL A,B,C;
        300 FILE ENTI (KIND=REMOTE),
        400      SALI (KIND=REMOTE);
        500 RE AD (ENTI,/,A,B,C);
        600 WRITE (SALI,("LAS VARIABLES QUE LEI",
        700    " FUERON:  ",3(/,F12.3)),
        800 A,B,C);
        900 END.
        #
[2]---> C N ALGOL
[3]---> #UPDATING
        #COMPILING 0270
[4]---> RE AD (ENTI,/,A,B,C);
                ERROR:UNDECLARED IDENTIFIER.     =RE
        #SNTX
[5]---> #ET=11.6 PT=0.7 IO=1.9
```

[1] : Se lista primero el archivo de trabajo. (Aunque esto no es necesario para la compilacion, pero si el tener presente el archivo de trabajo).

[2] : Se manda compilar. (En este caso particular fue necesario especificar el compilador ya que el archivo habia sido creado en el MAKE con otro tipo).

[3] : La maquina responde, primero, que esta actualizando el archivo de trabajo; despues, que ha comenzado a compilar.

[4] : A continuacion, escribe una lista de errores que encontro, poniendo primero el fragmento donde encontro el error y posteriormnete el tipo de error que la maquina considera que se cometio.

[5] : Por ultimo indica los tiempos utilizados en la compilacion.

Una vez que hemos corregido el programa, se intenta de nuevo compilarlo:

Ejemplo 2:

```
        #
[1]---> L
        100 BEGIN
        200 REAL A,B,C;
        300 FILE ENTI (KIND=REMOTE),
        400      SALI (KIND=REMOTE);
        500 READ (ENTI,/,A,B,C);
        600 WRITE (SALI,<"LAS VARIABLES QUE LEI",
        700    " FUERON:  "3(/,F12.3)>,
        800 A,B,C);
        900 END.
        #
[2]---> C W ALGOL
        %UPDATING
        %COMPILING 0305
        %ET=17.0 PT=1.1 IO=2.1
[3]---> R
        %RUNNING 0306
        ?
        1234
        2354.09
        8.8
        LAS VARIABLES QUE LEI FUERON:
             1234.000
             2354.090
                8.800
        %ET=32.0 PT=0.2 IO=1.8
```

[1] : Se lista nuevamente el archivo de trabajo, donde se puede ver que el error anterior ha sido corregido.

[2] : Se le manda compilar, a lo cual la maquina responde que actualiza, que compila y finalmente da los tiempos de maquina utilizados en la compilacion con lo que se puede asegurar que la sintaxis del programa es correcta.

[3] : Una vez compilado el programa, se hace que este corra por medio de un RUN, respondiendo la maquina que el programa

corre y pidiendo los datos, terminando posteriormente de correr el programa.

Como se puede observar, la corrida solo se puede realizar si la compilacion se ejecuto exitosamente, que es el caso del ultimo ejemplo.

OPCION 2:

RUN

Instruccion que en el caso de que la compilacion no fuese aun realizada, la ejecuta automaticamente pasando posteriormente a la corrida del programa.

Sin embargo, si el programa fuente tiene errores (de sintaxis) despues de darle a la maquina la instruccion COMPILE o RUN, no se podra crear el programa objeto y por lo tanto, no sera posible correr el programa, haciendo inutil el objeto de la instruccion RUN. En este caso la computadora lo indicara a traves de una lista de los errores que encontro, ante lo cual, el usuario debera corregir su archivo de trabajo para intentar de nuevo una compilacion.

Ahora bien; si el programa fuente tiene sus instrucciones correctas (sintacticamente), la computadora contestara con los tiempos utilizados en la compilacion y, naturalmente, no habra enunciado de errores, con lo cual sera posible la realizacion de la corrida.

Ejemplo:

```
E1]---> RUN
        #UPDATING
        #COMPILNG 3478
        #ET=10:0 PT=5.0 IO=4.8
        #RUNNING 5676
        #?
        1234
        2354.09
        8.8
        LAS VARIABLES QUE LEI FUERON:
              1234.000
              2354.090
                 8.800
        #ET=13:15 PT=5.90 IO=4.8
```

E1] : Se corre el programa una vez que se tiene la seguridad de que ya no tiene errores. Como aun no ha sido compilado, primeramente se realizara la compilacion para que inmediatamente despues se corra.

?TI, ?CS

Al compilar un programa, se da el caso de que se necesita saber acerca de la situacion de la compilacion en un momento dado, para lo cual contamos con las instrucciones de control llamadas ?TI y ?CS.

Con la primera, obtenemos informacion inmediata acerca de los tiempos de la maquina en un momento dado de la compilacion; mediante la segunda, la maquina informa acerca del numero de secuencia en que la compilacion se encuentra y ademas, de la cantidad de errores que ha encontrado hasta ese momento. Principalmente se utiliza para saber si ya se esta efectuando la compilacion. Se enuncian de la siguiente forma:

?TI

?CS

A continuacion se dan ejemplos en donde se usan estas instrucciones, una en cada caso.

Ejemplo 1:

```
                #
E1]---> C W ALGOL
        #COMPILING 0024
E2]---> ?CS
        SEQ=00000000, NO ERRORS
E3]---> ?CS
        INVALID NUMBER
        #ET=8.7 PT=0.8 IO=1.3
```

[1] : Inicialmente se ordena a la maquina que compile el programa a lo cual responde que ha comenzado a hacerlo.

[2] : Cuando se da la instruccion ?CS, la maquina responde con la secuencia en que se encuentra, y el numero de errores encontrados hasta ese momento (en realidad, la secuencia 0 indica que se esta por comenzar).

[3] : Al intentarse dar de nuevo esta instruccion, la maquina la desconoce puesto que para entonces, la compilacion ya ha terminado, (lo cual es indicado inmediatamente con los tiempos de maquina).

Ejemplo 2:

```
        #
[1]---> COMPILE WITH ALGOL
        #COMPILING 9864
[2]---> ?TI
        TIMES FOR 9864
          PROCESS =00:00:00 LIMIT 00:05:00
          IO      =00:00:00 LIMIT 00:10:00
          ELAPSED =00:00:10
[3]---> #ET=11.6 PT=0.8 IO=1.4
```

[1] : Se compila el programa.

[2] : Al dar la instruccion ?TI a la maquina, esta responde con los tiempos de maquina correspondientes a la compilacion con el numero de mezcla especificado.

[3] : Finalmente la maquina indica que la compilacion ha terminado.

?CS, ?STATUS, ?Y

Los comandos de control ?CS, ?STATUS e ?Y son propios de la consulta durante la compilacion pero son explicados mas adelante (despues del comando RUN) debido a que tambien se puede hacer uso de ellos durante la corrida. Sin embargo, se usan para el mismo fin y con la misma estructura en ambos casos.

RUN

El comando RUN tiene como funcion ejecutar o correr un programa. Los comandos RUN y EXECUTE realizan la misma funcion y es por ello que ambos provocaran la corrida de un programa al ser aplicados. Las opciones del comando RUN son:

OPCIONES:

1)RUN

2)RUN <Nombre del Archivo>

3)RUN; FILE <Archivo de Salida> (KIND=PRINTER)

OPCION 1:

RUN

Por medio de esta opcion que es la mas utilizada, se indica a la maquina que corra el programa que se tiene presente en el momento de usar el comando, o sea, el archivo de trabajo. Si al usar el RUN el programa no ha sido aun compilado, esta operacion sera ejecutada automaticamente, (siempre que no se necesite utilizar otro compilador). Una vez compilado el programa, la maquina ejecutara la corrida inmediatamente despues de dada la instruccion RUN.

Utilizando un mismo programa se muestran las diferentes opciones de RUN a traves de los siguientes ejemplos:


```
100 PROGRAM SUMA (INPUT, OUTPUT);
200 VAR
300       A,B,C:REAL;
400 BEGIN
500       READ (A,B);
600       C:=A+B;
700       WRITELN('A=',A,'B=',B,'C=',C);
800 END.
```

Ejemplo 1:

```
       #
[1]---> GET PASCAL1
[2]---> #HOPLFILE PASCAL1: ALGOL, 8 RECORDS, SAVED
[3]---> COMPILE WITH PASCAL
[4]---> #COMPILING 1579
       #ET=6.8 PT=1.4 IO=1.3
[5]---> RUN
[6]---> #RUNNING 1584
       #?
[7]---> 2.4142 3.1416
[8]---> A=    2.4142B=    3.1416C=    5.5558
[9]---> #ET=22.8 PT=0.4 IO=1.1
```

[1] : Primeramente se manda llamar el programa. El nombre con el que se encuentra guardado en la biblioteca es: PASCAL1.

[2] : La maquina contesta que existe y da su tamano.

[3] : Se le manda compilar (con el WITH puesto que es un programa en Pascal).

[4] : La maquina contesta que ha compilado.

[5] : Una vez compilado el programa, se teclea un RUN para correrlo.

[6] : La maquina contesta que lo esta corriendo y pide los datos.

[7] : Los datos le son dados.

[8] : La maquina ejecuta las operaciones correspondientes, y proporciona los resultados del programa.

[9] : A continuacion da los tiempos de maquina que indican que ha terminado de ejecutar el programa.

OPCION 2:

RUN <Nombre del Archivo>

Esta opcion sirve para indicar a la maquina que ejecute un programa que se encuentra guardado en el disco como un archivo de biblioteca. Si el programa objeto no esta guardado sera necesario compilar el programa fuente. La maquina hara la compilacion automaticamente al darsele la instruccion RUN, pero habra que realizarla "manualmente" si se desea especificar otro compilador. (Como es el caso de los programas en lenguaje Pascal).

Ejemplo 2:

```
        #
[1]---> RUN PASCAL1
[2]---> #RUNNING 1599
        #?
[3]---> 2.4142 3.1416
[4]---> A=    2.4142B=    3.1416C=    5.5558
[5]---> #ET=40.1 PT=0.4 IO=0.9
```

[1] : Se da la instruccion RUN con el nombre del archivo que se quiere correr.

[2] : La maquina indica que esta corriendo ese programa y pide los datos.

[3] : Se le proporcionan.

[5] : Y finalmente da los tiempos de maquina con lo que el programa termina de correr.

OPCION 3:

RUN: FILE <Archivo de Salida> (KIND=PRINTER)

Esta opcion produce que los resultados de la corrida de un programa no sean escritos en la terminal misma sino que sean mandados a la impresora en papel.

Para cada lenguaje se usa un tipo de instruccion dependiendo de la sintaxis de sus archivos:

Para Basic:

RUN: FILE BLINE (KIND=PRINTER)

Para Fortran:

RUN: FILE FILE6 (KIND=PRINTER)

Para Cobol y Algol:

RUN: FILE <Archivo de Salida> (KIND=PRINTER)

Para Pascal:

RUN: FILE OUTPUT (KIND=PRINTER)

En donde (Archivo de Salida) en Cobol y Algol es especificado por el usuario en su Programa.

Ejemplo 3:

```
        #
[1]---> GET PASCAL1
        #WORKFILE PASCAL1: ALGOL, 8 RECORDS, SAVED
[2]---> COMPILE WITH PASCAL
        #COMPILING 1630
        #ET=6.5 PT=1.4 IO=1.4
[3]---> RUN; FILE OUTPUT (KIND=PRINTER)
        #RUNNING 1557
        #?
[4]---> 23.4 34.6
[5]---> #ET=1.2 PT=0.8 IO=0.5
```

[1] : Se manda traer el archivo del disco.

[2] : Se le manda compilar, lo cual es ejecutado por la maquina.

[3] : Se corre el programa, especificando que se quiere que los resultados se escriban en impresora. La maquina responde que corre el programa y pregunta por los datos.

[4] : Los datos le son dados a la maquina.

[5] : Puesto que los resultados fueron escritos en la impresora, la maquina unicamente responde con los tiempos de maquina.

?END

Este comando tiene como funcion, el indicar el final de un conjunto de datos durante una corrida, con lo que la maquina dara por terminada la entrada de datos correspondiente. Como se puede observar esta instruccion es un comando de control, puesto que esta antecedida por el signo "?".

Esta instruccion se usa en cualquier programa que esta estructurado de tal forma que seguira pidiendo datos hasta que se de la instruccion ?END, la cual interrumpira inmediatamente la lectura de datos de entrada, y es por ello que podra ser utilizada para este efecto en otros casos.

El siguiente ejemplo aclara lo anterior:

Tomamos un Programa escrito en Fortran, puesto que es un lenguaje que se presta para representar el uso del ?END.

```
      100      5 READ(5,10)I,J,K
      200     10 FORMAT(3I2)
      300        LS=I+J+K
      400        WRITE(6,15)I,J,K,LS
      500     15 FORMAT("I=",I2,"J=",I2,"K=",I2,"LS=",I3)
      600        GO TO 5
      700        STOP
[1]---> 900      END
```

En el programa anterior, en la linea 600 hay un GO TO 5, o sea, una instruccion que ordena que nuevamente se lean mas datos; un ?END indicara el final de los datos y el Programa dejara de correr

[1] : No confundir el END de Fortran de fin de programa con el comando ?END.

```
        #
[1]---> RUN
        #RUNNING 3828
        #?
[2]---> 2 3 4
[3]---> I=2  J=13  K=4  LS=19
        #?
[4]---> 3 4 5
        I=3  J=4  K=5  LS=12
[5]---> ?END
[6]---> #ET=10.0 PT=0.9 IO=0.8
```

[1] : Se manda correr el programa, contestando la maquina que lo hace y pidiendo los datos.

[2] : Se proporcionan los datos a la maquina.

[3] : La maquina escribe el resultado del programa y vuelve a pedir mas datos.

[4] : Nuevamente se le proporcionan. (Este proceso podria seguirse indefinidamente). La maquina corre el programa y vuelve a pedir mas datos.

[5] : Esta vez se desea interrumpir la entrada de datos por lo que se le da un ?END. Dado que el programa no requiere mas datos, aqui termina la corrida.

[6] : La computadora responde con los tiempos de maquina con lo que se da por terminada la corrida.

?DS

El comando ?DS se utiliza para suspender o descontinuar una tarea o un programa (esta accion se ejecuta inmediatamente).

La utilizacion de este comando es necesaria cuando se tiene el problema de que el programa caye dentro de un proceso iterativo infinito; es decir, que si no se detiene va a gastar los recursos de la maquina innecesariamente (ya que caye dentro de un loop).

Otro de los casos en que se utiliza dicho comando, es cuando el proarama no esta realizando lo que se esperaba.

Tambien se utiliza este comando cuando el sistema esta muy saturado (esto es que muchos usuarios quieren entrar en alguna corrida o compilacion en el mismo instante en que el usuario quiere entrar).

O se tiene la necesidad de salirse de sesion porque se termino el tiempo en la terminal, por lo que se descontinua la espera a ejecucion del programa.

Con los siguientes ejemplos se muestran algunos de los casos en que se utiliza el comando ?DS :

Ejemplo 1:

```
[1]---> GET HIST1
[2]---> #WORKFILE HIST1: ALGOL, 32 RECORDS, SAVED
[3]---> COMPILE WITH PASCAL
        #COMPILING 2436
[4]---> ?OS
[5]---> #2436 OPERATOR DSED @ 276:0063:0
        #0-OS @ 44240S00, 44904000, 00000898.
        #ET=1:21.1 PT=1.1 IO=1.6
[6]---> SR
        #WORKSOURCE ALREADY SAVED
        BYE
        #END SESION 810 ET=21:35.3 PT:9.7 IO=8.9
        #USER = MC81 00:12:10 04/16/82
```

[1] : Se llama algun programa.

[2] : Indicando la maquina.

[3] : Una vez que se tiene presente, se compila.

[4] : Pero en el momento de la compilacion resulta que se termino el tiempo en la terminal, por lo que se descontinua la compilacion.

[5] : Indicando que la compilacion fue descontinuada.

[6] : Se salva el programa y se sale de sesion.

Ejemplo 2:

```
[1]---> GET HIST1
[2]---> #WORKFILE HIST1: ALGOL, 32 RECORDS, SAVED
[3]---> COMPILE WITH PASCAL
        #COMPILING 3159
        #ET=11.9 PT=0.8 IO=1.2
[4]---> RUN
[5]---> RUNNING 3454
        #3454 250 SECT REQ ON PACK PK117 ?
[6]---> ?OS
        #3454 OPERATOR DSED @ 276:0063:0
        #0-OS @ 44240S00, 44904000, 00000980.
        #ET=25.5 PT=0.7 IO=1.1
[7]---> SR
        #WORKSOURCE ALREADY SAVED
        BYE
        #END SESION 190 ET=35:15 PT=9.9 IO=5.9
        #USER = MC81 00:17:35 06/23/82
```

[1] : Se llama al programa.

[2] : Indicando la maquina.

[3] : Una vez que se tiene presente se compila.

[4] : Ya compilado, el programa se ejecuta.

[5] : Pero en la corrida indica que necesita recursos.

[6] : Por tal motivo se descontinua el programa ya que no es posible correrlo.

[7] : Una vez que se descontinuo el programa, se salva y se sale de sesion.

?STATUS

Este comando tiene como funcion informar al usuario el estado de una ejecucion (compilacion, corrida, etc) de un programa.

Dicho comando se utiliza cuando la respuesta a una ejecucion no es inmediata.

Esto sucede cuando el sistema esta muy saturado, es decir en ese momento muchos usuarios quieran compilar o ejecutar algun programa.

Al ver que la maquina no responde inmediatamente cuando se le manda alguno de los comandos RUN o COMPILE, para saber en que lugar de espera se encuentra el programa o para informarse si termino con el proceso, sera necesario teclearle el comando: ?STATUS.

Ejemplo 1:

```
[1]---> GET PASCAL/1
[2]---> #MODEL.FILE PASCAL/1: ALGOL, 9 RECORDS, SAVED
[3]---> COMPILE WITH PASCAL
        #COMPILING 1774
[4]---> #ET=25.4 PT=1.3 IO=1.3
[5]---> RUN
[6]---> #WAITING FOR AVAILABLE TASK
        ?STATUS
[7]---> #15:11 2ND IN TASK QUEUE
        #RUNNING 1435
        - - - -
        - - - -
        - - - -
[8]---> #ET=17:0 PT=0.9 IO=1.0
[9]---> SA
        #MODEL.SOURCE ALREADY SAVED
[10]---> BYE
        #END SESION 810 ET=20:15.6 PT=9.15 IO=7.89
        #USER = MC81 00:15:45 04/22/82
```


[1] : Se manda a llamar algun programa.

[2] : La maquina contesta.

[3] : Se compila dicho programa.

[4] : Contestando la maquina que ha terminado de compilar, sin haber detectado error alguno.

[5] : Inmediatamente se corre el programa.

[6] : Indica que el programa esta en espera para ser ejecutado. Si se desea preguntar en que lugar se encuentra el Programa o el estado del mismo, se teclea: ?STATUS.

[7] : Indicando la maquina que esta en segundo lugar en la cola de espera para ejecucion.

[8] : Pasado algun tiempo, la maquina dara por terminado el proceso, indicando los tiempos.

[9] : Una vez terminada la ejecucion, se salva el programa.

[10] : Ya que se tiene salvado el programa, se sale de sesion.

Ejemplo 2:

```
[1]---> GET TAREA/2
[2]---> #WORKFILE TAREA/2: ALGOL, 18 RECORDS, SAVED
[3]---> RUN
[4]---> #WAITING FOR AVAILABLE TASK
[5]---> ?STATUS
[6]---> #15:11 2ND IN TASK QUEUE
[7]---> ?STATUS
[8]---> #COMPILING 4567
        #15:12 ET=5:38 PT=0.0 IO=0.0
        #ET=33.1 PT=0.4 IO=1.0
[9]---> #WAITING FOR AVAILABLE TASK
[10]---> ?STATUS
[11]---> #15:15 3RD IN TASK QUEUE
[12]---> ?STATUS
        #15:15 3RD IN TASK QUEUE
        ?STATUS
        #15:16 3RD IN TASK QUEUE
        ?STATUS
        #15:16 2ND IN TASK QUEUE
        ?STATUS
        #15:17 2ND IN TASK QUEUE
        ?STATUS
        #15:17 2ND IN TASK QUEUE
        #RUNNING 4060
        - - - -
        - - - -
        - - - -
[13]---> #ET=1:03.4 PT=0.3 IO=1.0
[14]---> ?STATUS
[15]---> #NOT BUSY
[16]---> SR
        #WORKSOURCE ALREADY SAVED
        BYE
        #END SESION 567 ET=23:25 PT=12.8 IO=8.3
        #USER = MC81 00:13:13 04/23/82
```


[1] : Se manda a llamar algun programa.

[2] : Contestando la maquina.

[3] : Se manda a correr sin antes compilarlo, en el momento que la maquina esta un poco saturada de tareas queriendo compilar o correr.

[4] : Contestando la maquina.

[5] : Pero como el programa aun no ha sido compilado, la cola de espera en la que se encuentra el programa, es para compilar. Al ver que no se recibe respuesta, se teclea el comando: ?STATUS.

[6] : Contestando la maquina.

[7] : Nuevamente se teclea el comando.

[8] : Se detiene un momento y se observa que la maquina manda los tiempos, lo que indica que termino de compilar.

[9] : Inmediatamente contesta lo que indica que el programa esta en cola de espera para ser ejecutado.

[10] : Para informar que lugar esta ocurando el programa se teclea: ?STATUS.

[11] : Contestando.

[12] : Se teclea el comando varias veces mas.

[13] : Se deja de teclear y un momento mas tarde la maquina contesta con los tiempos de maquina, lo cual indica que termino de ejecutar el programa.

[14] : Si nuevamente se teclea el comando, se tiene.

[15] : Contestando la maquina que no tiene nada que informar.

[16] : Por lo tanto se salva el programa y se sale de sesion.

El ejemplo anterior se refirio al caso en que la tarea se encuentre en cola, pero cuando la maquina se encuentra realizando alguna ejecucion (como una compilacion o una corrida), es posible preguntarle acerca de la situacion en que se encuentra esta actividad por medio de la instruccion de control (este tipo de instrucciones se caracterizan por su capacidad de controlar e interrogar al sistema operativo, cuya respuesta es inmediata) llamada ?STATUS. Ante esta instruccion, la maquina generalmente responde con los tiempos de maquina de la actividad que realiza, lo cual indica que no esta en cola de espera para compilar o ejecutar.

Ejemplo:

```
[1]---> RUN
        #RUNNING 4968
        #?
[2]---> ?STATUS
        #12:12 ET=7.1 PT=0.1 IO=1.0
        ?STATUS
        #12:12 ET=17.6 PT=0.1 IO=1.0
        ?STATUS
        #12:12 ET=41.5 PT=0.2 IO=1.0
[3]---> 7486574
        7487593
[4]---> LAS VARIABLES QUE LEI:
        7486574.000
        7487593.000
        #ET=54.9 PT=0.3 IO=1.1
[5]---> ?STATUS
[6]---> #12:15 NOT BUSY
```

[1] : Si se tiene un programa ya compilado, se manda a correr. Inmediatamente despues pedira los datos.

[2] : En el mismo instante se pregunta por la situacion de la tarea, varias veces.

[3] : Se le proporcionan los datos a la maquina.

[4] : Dando como resultados.

[5] : Una vez que ha terminado de correr el programa, se le pide informacion del estado del programa.

[6] : Contestando que este comando no procede cuando no se esta ejecutando alguna tarea.

?Y

Este comando proporciona la informacion acerca de las caracteristicas de una corrida o compilacion.

Es decir, con este comando se tiene la oportunidad de informarse de varios datos o caracteristicas que el sistema utiliza internamente en una corrida o compilacion como son :

a) Estado del proceso (a la hora de teclear el comando).

b) Prioridad.

c) Origen de la transmision LSN (Logical Station Number).

d) Estado del Stack.

e) Nombre del compilador.

f) Nombre del programa.

g) Ocasionalmente algun evento extraordinario que puede ser :

-Utilizacion de archivos que no estan en el PACK.

-Espera de asignacion de recursos, etc.

Este comando es utilizado cuando en una corrida o compilacion se requiere la informacion del estado del programa dentro del procesador, es decir que mas necesita (asignacion de memoria, asignacion de algun otro archivo, etc) o simplemente si permanece aun activa.

Los siguientes ejemplos mostraran lo explicado anteriormente :

Ejemplo 1:

```
[1]---> GET PASCAL/EJEMPLO
[2]---> #MODIFILE PASCAL/EJEMPLO: ALGOL, 9 RECORDS, SAVED
[3]---> COMPILE WITH PASCAL
[4]---> #WAITING FOR AVAILABLE TASK
[5]---> #COMPILING 3420
[6]---> ?Y
[7]---> STATUS OF TASK 3420 AT 15:09:32
        PRIORITY = 50
        STACK STATE: ALIVE
        COMPILER NAME: SYSTEM/PASCAL
        PROGRAM NAME: (MCBI)CANDE/CODE350 ON PACK
        #ET=9.0 PT=2.9 IO=1.8
[8]---> ?Y
        #NOT BUSY
```

[1] : Se llama un programa que se encuentre en la biblioteca del usuario.

[2] : Indicando la maquina.

[3] : Se compila el programa.

[4] : Indicando que el sistema se encuentra un poco saturado.

[5] : Despues de algun tiempo se recibe la senal de que la compilacion se esta realizando.

[6] : En este momento se requiere informacion del historial del programa, para lo cual se teclea el comando ?Y.

[7] : Contestando la maquina con toda la informacion relacionada con la compilacion del programa, y que en en este caso se refiere a que la compilacion esta activa.

[3] : Una vez que la maquina contesto que el programa ha sido compilado, no se puede volver a teclear el comando, pues este solo se utiliza cuando existe alguna tarea en ejecucion.

Ejemplo 2:

```
[1]---> RUN
[2]---> %WAITING FOR AVAILABLE TASK
[3]---> RUNNING 3421
[4]---> ?Y
[5]---> STATUS OF TASK 3421 AT 15:01:58
        PRIORITY = 50
        ORIGINATION: LSN 35
        STACK STATE: WAITING ON AN EVENT
        PROGRAM NAME: (MCS1)CANDE/CODE350 ON PACK
        RSVP: NO FILE OVERSEAS/ROC/DATOSNES ON PACK ELSE DISK
        REPLY: FA,UL,IL,OK,DS
```

[1] : Se corre el programa.

[2] : Se indica que el sistema esta un poco saturado.

[3] : Mas tarde contesta que el programa esta corriendo.

[4] : Pero el usuario quiere el historial de su programa en este momento, por lo cual se teclea el comando ?Y para que el sistema indique los datos requeridos.

[5] : Contestando la maquina con la informacion relacionada con la corrida del programa. En este caso se refiere a que se trata de utilizar algun archivo que no se encuentra en el PACK y pide alguna respuesta entre la que podriamos dar:

?DS : Descontinua la corrida.

?FA KIND=OTROPACK : Asigna el PACK OTROPACK y busca en este el archivo que se necesita para continuar la corrida.

Descripcion de las caracteristicas que el sistema utiliza internamente:

Para esto haremos uso del ejemplo 1:

?Y

STATUS OF TASK 3420 AT 15:09:32

Indica el estado de la tarea o programa que esta realizando en el momento en que se teclea el comando ?Y. El numero 3420 es el numero de mezcla que la maquina le dio automaticamente al programa, este numero indica el numéro de procesos en la cola que ha realizado la maquina antes de ejecutar el programa.

PRIORITY = 50

Indica que el programa se encuentra en prioridad 50. La prioridad es una caracteristica que el sistema utiliza para determinar que programas corren primero que otros.

ORIGINATION: LSN 35

Estacion que transmite la informacion a la maquina (que terminal o teletipo manda dicha informacion) LSN significa: Logical Station Number.

STACK STATE: ALIVE

Estado del STACK en el momento de que tecleamos el comando ?Y. En este caso nos indica que el STACK esta activo sin problema alguno.

COMPILER NAME: SYSTEM/PASCAL

Nombre del compilador con el que se esta trabajando.

PROGRAM NAME: (MCBI)CANDE/CODE350 ON PACK

Nombre del programa con el que se esta trabajando, y ademas la clave del usuario.

COMANDOS PROPIOS DE FIN DE SESION

SAVE

Este comando guarda archivos en disco. Se usa cuando se desea que un archivo de trabajo pase a formar parte del directorio del usuario, o bien cuando se quiere que dicho archivo de trabajo sustituya a una version antigua en el directorio.

OPCIONES:

1) SAVE

2) SAVE AS <Nombre de Archivo>

OPCION 1:

SAVE

El comando SAVE se utiliza para proteger archivos de trabajo guardandolos en disco.

Esta instruccion guarda el archivo de trabajo y su objeto asociado (en caso de estar presente) en la biblioteca del usuario, dicho archivo podra llamarse posteriormente con el nombre con el que se creo (ver comando GET).

Si en el momento de salvar un archivo de trabajo por medio de SAVE existia ya uno en la biblioteca con el mismo nombre, este

ultimo sera removido.

Una vez salvado, el archivo de trabajo quedara presente y podra modificarse o ejecutarse si se continua en sesion. en caso de entrar a una nueva sesion se mandara traer para su modificacion o ejecucion (ver GET).

Es importante volver a hacer enfasis en el hecho de que al salvar un archivo (ya sea programa, datos, etc.) el sistema borrara las versiones antiguas dejando unicamente la mas reciente. esto es debido al hecho de que solo se admite en la biblioteca del usuario un solo archivo con el mismo nombre. Por lo tanto si se desea guardar la version anterior y tambien la nueva, se debera guardar con un nombre diferente haciendo uso de la opcion 2.

NOTA:

. El hecho de que en la biblioteca del usuario solo puede existir un archivo por nombre habra que tenerlo muy en cuenta cuando se usa una clave compartida.

Ejemplo 1:

```
[1]---> M EJEMPLO FORTRAN
        #WORKFILE EJEMPLO.FORTRAN
        SEQ
        100         READ(5,*)A,B
        200         SUMA=A+B/A
        300         WRITE(6,*)A,B
        400         WRITE(6,*)SUMA
        500         CALL EXIT
        600         END
        700
        #
[2]---> SA
[3]---> #UPDATING
        #WORKSOURCE EJEMPLO SAVED
```

[1] : Primeramente se crea un archivo con el nombre de EJEMPLO (ver comandos MAKE y SEQ).

[2] : Usando la instruccion SA el archivo EJEMPLO queda guardado en disco.

[3] : Esta respuesta significa que el archivo fuente (WORKSOURCE) unicamente quedo guardado, en el caso del objeto no fue así debido a que, como nó se ha compilado, no existe aun.

Si se da el comando SAVE unicamente y existe tanto el archivo fuente como el objeto, ambos seran salvados como puede verse en el siguiente ejemplo:

Ejemplo 2:

```
          M CANDE/EJEMPLO FORTRAN
          #100) FILE CANDE/EJEMPLO;FORTRAN
          SEQ
          100         READ(5,/)A
          200         WRITE(6,/)A
          300         STOP
          400         END
          500
          #
  [1]---> COMPILE
          #FORTING
          #COMPILING 1006
          #ET=5.2 PT=0.8 IO=1.3
  [2]---> SA
  [3]---> #WORKFILES CANDE/EJEMPLO SAVED
```

[1] : Se compila el programa CANDE/EJEMPLO para que se genere un archivo objeto (de trabajo, o sea WORKOBJECT).

[2] : Se teclea SA.

[3] : Los archivos fuente y objeto quedan salvados (notese que la terminal contesta "WORKFILES" en vez de "WORKSOURCE" o "WORKOBJECT" como en el ejemplo anterior, esto es debido, como ya se menciono, a que quedan salvadas las dos versiones del archivo de trabajo).

Y el directorio quedaria asi:

```
(MCS1) ON PACK
  CANDE
    EJEMPLO : FORTRAN
  *OBJECT
    CANDE
      EJEMPLO : FORTRANCODE
```

OPCION 2:

SAVE AS (Nombre de Archivo)

La opcion SAVE AS (Nombre de Archivo) sirve para salvar un archivo de trabajo y/o su archivo objeto bajo un nombre diferente. Si ya existe el objeto (es decir si ya fue compilado, ver COMPILE) entonces este quedara salvado bajo el nombre: OBJECT/(Nombre de Archivo).

Si ya existe un archivo en la biblioteca del usuario bajo el nombre con el que se desea salvar un archivo de trabajo, el comando sera rechazado.

Ejemplo 1:

```
[1]---> LIST
        100         READ(5,*)A,B
        200         SUMA=A+B+A
        300         WRITE(6,*)A,B
        400         WRITE(6,*)SUMA
        500         CALL EXIT
        600         END
        #
[2]---> C
        #COMPILING 8370
        #ET=6.4 PT=0.9 IO=1.3
[3]---> SA SOURCE AS F5/EJ
        #100 SOURCE EJ SAVED AS (JPS1)F5/EJ ON PACK
[4]---> SA OBJECT AS F5/EJ
        #100 OBJECT EJ SAVED AS (JPS1)OBJECT/F5/EJ ON PACK
[5]---> FILES F5
        (JPS1) ON PACK
          F5
            EJ : FORTRAN
        #
        FILES OBJECT/F5
        (JPS1) ON PACK
          OBJECT
            F5
              EJ : FORTRANCODE
        #
```

[1] : Se lista el archivo de trabajo, que en este caso es un programa en FORTRAN.

[2] : Se manda compilar para producir el programa OBJETO.

[3] : La instruccion SA SOURCE AS F5/EJ produce una "copia" del archivo de trabajo (unicamente de la version fuente) y la guarda en disco bajo el nombre F5/EJ; lo anterior se resume en la respuesta que da la terminal.

[4] : Con la instruccion SA OBJECT AS F5/EJ se guarda unicamente el objeto de nuestro archivo de trabajo. Obsérvese que, como se infiere de la respuesta de la terminal, el objeto queda guardado bajo el nombre OBJECT/F5/EJ poniendo el directorio OBJECT/ sin necesidad de indicarselo, esto es debido a que todos los objetos quedan guardados en el disco bajo el directorio OBJECT/, siendo esta la forma en que la computadora reconoce si ese archivo es OBJETO o FUENTE.

[5] : Finalmente se usa el comando FILES F5 (ver comando FILES) para la obtencion de todos los archivos guardados bajo el directorio F5 viendo en la lista el archivo que se acaba de guardar bajo el nombre F5/EJ. Igualmente, con FILES OBJECT/F5 se pregunta por los OBJETOS guardados bajo el directorio F5 viendo que el archivo objeto OBJECT/F5/EJ quedo guardado como en el caso anterior.

Ejemplo 2:

Suponiendo que se tiene como archivo de trabajo un archivo llamado LEE:

```
        L
        100 READ(5,/)N
        200 STOP
        300 END
        #
[1]---> SAVE SOURCE AS FORT
        #WORK.SOURCE LEE SAVED AS (MC81)FORT.ON PACK
[2]---> SAVE OBJECT AS FORT
        #NO WORK.OBJECT
[3]---> C
        #COMPILING 0531
        #ET=0.6 PT=0.5 IO=1.2
[4]---> SAVE OBJECT AS FORT
        #WORK.OBJECT LEE SAVED AS (MC81)OBJECT/FORT ON PACK
[5]---> FILES FORT
        (MC81) ON PACK
          FORT : FORTRAN
        #
        FILES OBJECT/FORT
        (MC81) ON PACK
          OBJECT
            FORT : FORTRANCODE
        #
```

[1] : Uso de la opcion SOURCE AS con el objeto de que quede almacenado el programa fuente bajo el nombre FORT, observese que la respuesta de la terminal indica claramente el resultado de la operacion.

[2] : Aqui se usa la opcion SAVE OBJECT AS para guardar el archivo objeto bajo el nombre FORT pero la terminal responde que no existe tal ("NO WORKOBJECT") por lo tanto es necesario:

[3] : Compilarlo para generar el archivo objeto.

[4] : Una vez mas se usa la opcion SAVE OBJECT AS observando de la respuesta que dicho objeto ha sido guardado bajo el nombre OBJECT/FORT, el directorio OBJECT/ es puesto para diferenciar entre el programa fuente y el objeto por la maquina, el usuario no tiene necesidad de ponerlo.

[5] : Para ver que efectivamente los archivos de trabajo(fuente y objeto) han sido guardados con los nuevos nombres, se presunta por ellos mediante FILES (recuerdese que FILES (nom. de arch.) lista unicamente el archivo por el que se presunta) viendo que efectivamente estos fueron guardados en la biblioteca del usuario.

Si unicamente se usa la opcion SA AS... quedaran guardados bajo el nuevo nombre ambos archivos (fuente y objeto) si es que los dos existen.

REMOVE

La instruccion REMOVE se usa para borrar archivos de trabajo, archivos de biblioteca y directorios completos de la biblioteca del usuario.

OPCIONES:

1) REM

2) REM <Tipo de Archivo>

OPCION 1:

REM

Si unicamente la instruccion REM es tecleada, solo el archivo de trabajo (o el objeto, en caso de existir) sera borrado.

A continuacion se muestra un ejemplo en el cual el archivo de trabajo EJEMPLO es removido.

```
[1]---> G EJEMPLO
        #WORK FILE EJEMPLO, FORTRAN, 6 RECORDS, SAVED
[2]---> LIST
        100         READ(5,*)A,B
        200         SUMA=A+B/A
        300         WRITE(6,*)A,B
        400         WRITE(6,*)SUMA
        500         CALL EXIT
        600         END
        #
[3]---> REM
        #
[4]---> LIST
        #NO WORK FILE
[5]---> WHAT
        #NO WORK FILE
```

[1] : Se llama EJEMPLO.

[2] : Se manda listar (ver LIST).

[3] : Mediante REM se borra. Al borrarlo, la terminal responde con el signo "#" (por ser archivo de trabajo, despues de borrado ya no se tendra presente pero el archivo EJEMPLO seguira existiendo en la biblioteca del usuario).

[4] : Se manda listar, respondiendo la terminal que no hay archivo presente.

[5] : Mediante WHAT (ver comando WHAT) se pregunta por el archivo de trabajo, respondiendo que no hay tal ("NO WORK FILE").

OPCION 2

REN <Tipo de Archivo>

Mediante esta opcion es posible renover del disco archivos fuente u objeto por separado por medio de los comandos REN SOURCE y REN OBJECT

En los siguientes ejemplos se muestra como pueden borrarse archivos fuente y objeto por separado.

```
[1]---> FILES FORTR
[2]---> (JFS1) ON PACK
        .  FORTR
        .  .  EJEMPLO : FORTRAN
        #
[3]---> FILES OBJECT/FORTR
        (JFS1) ON PACK
        .  OBJECT
        .  .  FORTR
        .  .  .  EJEMPLO : FORTRANCODE
        #
[4]---> REM OBJECT/FORTR/=
[5]---> # 1 FILE IN (JFS1)OBJECT/FORTR/= REMOVED ON PACK
[6]---> REM SOURCE FORTR/=
        # 1 FILE IN (JFS1)FORTR/= REMOVED ON PACK
        #
[7]---> FILES FORTR
        #NO FILE(S) ON PACK
        #
```

[1] : Primeramente se pregunta por el contenido del directorio FORTR.

[2] : La respuesta indica que solo hay un archivo, FORTR/EJEMPLO.

[3] : Se hace lo mismo que la instruccion anterior pero con el directorio OBJECT/FORTR la terminal responde que hay un objeto, el correspondiente a FORTR/EJEMPLO.

[4] : Con la instruccion OBJECT/FORTR/= se borra el archivo objeto

FORTR/EJEMPLO. La OPCION "/=" significa que lo que se quiere borrar son todos los archivos que estan bajo dicho directorio, en este caso solo hubo uno.

[5] : La terminal responde al comando anterior diciendo que habia un solo archivo bajo el directorio FORTR y este fue removido.

[6] : Con esta instruccion se borra la version fuente de los archivos que estan bajo el directorio FORTR contestando la terminal, al igual que el caso anterior, que solo existia un archivo fuente bajo ese directorio y fue removido.

[7] : Finalmente se pregunta por el contenido del directorio FORTR respondiendo la terminal que no hay archivos bajo este directorio.

BACKUPPROCESS

El comando BACK permite ver y manejar la informacion existente en el disco de impresion mediante las opciones NEXT, REMOVE, WHAT, COPY, LIST. Para un tratamiento mas detallado de las opciones anteriores se puede consultar el MANUAL CANDE B6700 y B7700.

Supongase que se acaba de realizar una corrida en donde el archivo de salida se ha declarado o se ha tomado por default como PRINTER. Siendo este el caso, se puede estar interesado en ver que es lo que se imprimira en papel, es decir que informacion esta en cola de impresion. Para esto, se usara el comando BACK de la siguiente forma:

1) Se tecleara el comando BACK.

2) Se dara alguna de las opciones mencionadas dependiendo de lo que se quiera hacer con la informacion obtenida por medio de BACK.

OPCIONES:

1) NEXT

2) REMOVE

3) WHAT

4) LIST

OPCION 1:

NEXT

La maquina muestra el siguiente archivo en la cola de impresion (esto servira cuando en una sesion se han hecho varias impresiones).

OPCION 2:

REMOVE

Borrara el archivo y ya no se mandara imprimir.

OPCION 3:

WHAT

Muestra el tamano del archivo, indica si fue producto de un Job, el nombre del archivo que le dio origen, etc.

OPCION 4:

LIST

Lista el archivo tal y como saldra impreso.

Ejemplo 1:

```
[1]---> L
        100         READ(5,*)A,B
        200         SUMA=A+B.'A
        300         WRITE(6,*)A,B
        400         WRITE(6,*)SUMA
        500         CALL EXIT
        600         END
        #
[2]---> R.FILE FILE6(PRINTER)
[3]---> #RUNNING 8384
        #?
        2,3
        #ET=14.8 PT=0.5 IO=1.2
[4]---> BACK
        #RUNNING 8385
[5]---> #?
        #ENTER 'NEXT','LIST','WHAT','REMOVE' OR 'COPY'
[6]---> #FILE BD:0008379/0008384/000FILE6 ON DISCO; 5 RECORDS
[7]---> LIST
[8]---> 100
        200
        300 2.0, 3.0,
        400
        500 3.5.
        #FILE BD:008379/0008384/000FILE6 ON DISCO; 5 RECORDS
[9]---> NEXT
        #
```

[1] : Se lista el archivo de trabajo.

[2] : Se corre el programa declarando el archivo de salida como PRINTER (Esto se hace para que los resultados se impriman en papel en las impresoras del CSC, ver comando RUN).

[3] : Corrida del programa; 2,3 son los datos.

[4] : Primer paso; comando BACK.

[5] : La terminal ofrece la lista de opciones y pide (con ?) la opcion que se desee.

[6] : Se muestra el archivo BACKUP existente en disco, en este caso dicho archivo tiene 5 registros.

[7] : Se da en primer lugar la opcion LIST, es decir, se quiere

listar el archivo, (segundo paso).

[8] : Archivo BACKUP listado por la terminal, este archivo consiste en este caso en los resultados de la ejecucion del programa que se corrio anteriormente, es decir, lo que se imprimira en papel al terminar la sesion, inmediatamente despues de listado el archivo aparecera otra vez su especificacion, notese que en esta especificacion aparece en segundo termino el numero de la corrida (8384 en este caso) que produjo ese archivo de impresion (ver [2]).

[9] : A continuacion se da la orcion NEXT para mostrar (si es que existe) el siguiente archivo, como solo se corrio un programa, solo habra un archivo, por lo tanto la terminal contesta con un "#" indicando que ya no hay archivos y se salio del BACK.

Ejemplo 2: Uso de la opcion REMOVE (esta opcion es importante cuando, por medio de la opcion LIST, se ha descubierto que los resultados son erroneos, con la instruccion REM el archivo es removido del disco de impresion y no saldra listado).

```
[1]---> BACK
        #RUNNING 8386
[2]---> #?
        #ENTER 'NEXT', 'LIST', 'WHAT', 'REMOVE' OR 'COPY'.
        #FILE BD:0008379/0008384:000FILE6 ON DISCO: 5 RECORDS
[3]---> REM
[4]---> #8386 BD:0008379/0008384:000FILE6 REMOVED ON DISCO FK118
        #
[5]---> BACK
        #RUNNING 8387
[6]---> #NO BACKUP FILES FOUND
        #
```

[1] : Se da la instruccion BACK.

[2] : Se muestra la lista de opciones y el archivo, en este caso es el mismo que en el ejemplo anterior, ya que tiene el mismo numero de corrida (8384).

[3] : Se da la instruccion REM (REMOVE).

[4] : La terminal contesta que el archivo producido por la corrida con numero de mezcla 8384 (BD:0008379/0008384) cuyo nombre es FILE6 (al correr el programa se declaro FILE6 como PRINTER, como usualmente se hace en FORTRAN), fue removido del disco de impresion.

[5] : Se da una vez mas la instruccion BACK para comprobar que el archivo fue removido.

[6] : La terminal contesta que no se encuentra ningun archivo para imprimirse ("NO BACKUP FILES FOUND").

NOTA: Ver comando RUN para conocer nombres de archivos de impresion.

SPLIT

El comando SPLIT termina la sesion y empieza una nueva con la misma clave del usuario y dejando presente y sin alterar el archivo de trabajo.

EJEMPLO:

```
[1]---> SPLIT
[2]---> #SPLIT SESSION 8366 ET=15:06.7 PT=7.5 IO=6.3
[3]---> #NEW SESSION 8379 18:38:14   04/24/82
[4]---> ?WRU
[5]---> #66700:1268 CANDE 31,280: YOU ARE GEOFISICA4(37)
        #SESSION = 8379   USER=JP81
```

[1] : Se teclea comando SPLIT.

[2] : Sesion que termina:8366 (ademas aparecen los tiempos cargados al usuario: transcurrido, procesador y E/S).

[3] : Sesion que empieza:8379 (tambien aparecen la hora y la fecha en que empieza la sesion).

[4] : La instruccion ?WRU sirve para conocer la terminal en que se trabaja y las caracteristicas de la sesion (ver comando ?WRU).

[5] : Caracteristicas de la sesion actual (ver comando ?WRU para mas detalles sobre la contestacion al comando).

BYE

El comando BYE termina la sesion y genera los tiempos cargados al usuario:

ET= Tiempo transcurrido entre el inicio y el fin de la sesion.

PT= Tiempo de CPU utilizado en la sesion.

IO= Tiempo de E/S (entrada/salida) utilizado en la sesion.

Cuando se tiene un archivo de trabajo recien creado, o uno corregido o compilado y la nueva version permanece sin salvar, el comando BYE sera rechazado y aparecera el siguiente mensaje:

REMOVE OR SAVE WORKFILE

Que recuerda al usuario salvar su archivo de trabajo creado o corregido.

Ejemplo:

```
[1]---> BYE
[2]---> #REMOVE OR SAVE WORKFILE
[3]---> REM
        #
[4]---> BYE
[5]---> #END SESSION 8379 ET=18:28.8 PT=7.8 IO=7.7
        #USER = JPSI 18:56:43  04/24/82
```

[1] : Se teclea comando BYE para terminar la sesion.

[2] : El comando se rechaza, aparece recordatorio.

[3] : Se remueve el archivo de trabajo.

[4] : Se teclea comando BYE .

[5] : Termina la sesion 8379, se indican ademas: (respectiva

mente).

1) Tiempos de ET, PT y IO en segundos.

2) Clave del usuario.

3) La hora y fecha en que termino la sesion.

APENDICE

RECOVERY

En algunas ocasiones el sistema sufre fallas y se corre el riesgo de perder los archivos de trabajo. Para prevenir estas situaciones, la maquina esta programada para proteger tales archivos, guardandolos en otro archivo especial que se llama RECOVERY.

Sin embargo, es necesario sacarlos de RECOVERY para poder seguir trabajando con ellos.

La instruccion RECOVERY sirve tanto para preguntar si existe algun archivo en el RECOVERY como para liberarlo y poderlo convertir en archivo de trabajo; por lo tanto, se tienen las siguientes opciones para la instruccion:

OPCIONES:

1) RECOVERY

2) RECOVERY (Numero Especificado)

OPCION 1:

RECOVERY

Mediante esta opcion se obtiene la informacion acerca de los archivos existentes en el archivo RECOVERY.

Ejemplo 1:

```
[1]---> REC
[2]---> #RECOVERY DATA ON UNAM1
[3]---> 440 EJEMPLO (04/20/82)
        690 ESM/COBOL (04/19/82)
        1030 INVITACIONES (03/07/82)
        #
```


[1] : Se manda la instruccion RECOVERY para ver los archivos que se tienen.

[2] : La maquina contesta que si se tienen archivos en RECOVERY en UNAM1.

[3] : Asimismo envia el numero de RECOVERY, el nombre del archivo y la fecha de creacion.

OPCION 2:

RECOVERY (Numero Especificado)

Con esta opcion se extraen los archivos del RECOVERY, lo cual es necesario para poder trabajar con ellos.

Ejemplo 2:

```
[1]---> REC 440
[2]---> #WORKFILE EJEMPLO; SEQ, 6 RECORDS, SAVED
```

[1] : Se indica el comando RECOVERY (REC) y el numero de RECOVERY del archivo; este numero se sabra efectuando una instruccion identica a la que se explica anteriormente.

[2] : La maquina indicara que se tiene presente el archivo de trabajo, el nombre del archivo y asimismo indicara que tiene secuencia y el numero de registros.

Despues de haber recobrado el archivo, ya no es necesario traerlo con el comando GET, puesto que ya se tiene y se puede trabajar con el en forma normal.

DISCARD

El comando DISCARD permite borrar un archivo que se encuentre en RECOVERY. Para ello hay que conocer el numero que el RECOVERY le asigna ya que sera asi la forma en que se podra identificar al ser borrado.

OPCIONES:

1) DISCARD <Numero Asignado>

Ejemplo :

```
DISCARD 448
#
```

Despues de haber efectuado el DISCARD, si se desea saber que archivos se tienen en RECOVERY, se observara que el archivo que borramos ya no aparece.

FILES

El comando FILES se usa para hacer que en la terminal se liste, el directorio completo del usuario o parte de el por medio de sus distintas opciones.

OPCIONES:

1) FILES

2) FILES <Nombre del Directorio>

3) FILES <Nombre del Archivo>

4) FILES <Numero de Profundidad>

OPCION 1:

FILES

Si unicamente se usa FILES, se listaran todos los archivos y directorios que esten en la biblioteca del usuario y el tipo al que pertenece cada uno.

Ejemplo 1:

```
[1]---> FILES
[2]---> (MCS1) OH PACK
        . OH : DATA
        . DS : DATA
        . ESM
        .   . FILA : ALGOL
        . DIRR
        .   . COLA : ALGOL
        .   . PILA : ALGOL
        .   . AUTOS : ALGOL
        .   . ORDENA : SEGDATA
        .   . CREDATO : SEGDATA
        #
```

[1] : Comando FILES sin orciones.

[2] : Se listan todos los archivos del usuario.
Se listan tambien los tiros.

Notese que los archivos ararecen en orden creciente según el numero de letras que tenga el nombre del archivo, y entre las que tienen el mismo numero de letras, ararecen ordenados por el orden alfabetico. Cada archivo va seguido, como ya se dijo, del tiro al que rertenecen (FORTRAN, ALGOL, COBOL, SER, DATA, etc.).

OPCION 2:

FILES (Nombre del Directorio)

Si solo se quieren listar los archivos que estan bajo un directorio en rarticular se usara FILES seguido por la opcion (Nombre del Directorio). (Es decir, solo interesa una determinada subfamilia).

Ejemplo 2:

```
[1]---> FILES PASCAL
[2]---> (MCBI) ON PACK:
        .  PASCAL
        .  .  1 : ALGOL
        .  .  2 : ALGOL
        .  .  3 : ALGOL
        .  .  PROG : ALGOL
        .  .  PROG* : ALGOL
        .  .  REBECA
        .  .  .  T1 : ALGOL
        .  .  TAREA1 : ALGOL
        .  .  TAREA2 : ALGOL
        #
```


[1] : Comando FILES seguido de la orcion (Nombre del Directorio), en este caso PASCAL es el nombre del directorio.

[2] : Archivos que estan bajo el directorio PASCAL.

OPCION 3:

FILES (Nombre del Archivo)

Si se usa la orcion (Nombre del Archivo) despues del comando FILES la terminal unicamente listara dicho archivo y el tiro al que rertenece.

Ejemplo 3:

```
[1]---> FILES FORTRAN1
[2]---> (MCBI) ON PACK:
        .  FORTRAN1 : SEQDATA
        #
```

[1] : Comando FILES seguido de un nombre de archivo, en este caso FORTRAN1.

[2] : La terminal listo el archivo y su tiro.

OPCION 4:

FILES (Nivel de Profundidad)

Tambien se rueden listar los archivos del directorio que tengan un nivel de rrofundidad (ver directorios) haciendo uso de la orcion, FILES seguido de ";" y un entero mayor que cero que indica el (Nivel de Profundidad) de los archivos que se listen.

Ejemplo 4:

```
[1]---> FILES 1
[2]---> (MICRO) ON PACK
        . DOS : DATA
        . DOS : DATA
        . ESM
        . DIRS
        . MPGC
        . ARIES
        . COBOL : SEQDATA
        . DATOS : DATA
        . DOBLE : DATA
        . HIST1 : ALGOL
        . HIST2 : ALGOL
        . NELLY
        . OBJECT
        . PASCAL
        . PRUEBA
        . ARCHIVO
        . ESCRIBE : DATA
        . PASCAL1 : ALGOL
        . FORTRAN1 : SEQDATA
        . FORTRAN2 : SEQDATA
        . PASCALES
        . COMPRUEBA
        . MARGENES2 : DATA
        #
```

[1] : Comando FILES seguido de la profundidad deseada. En este caso se desea que se liste el directorio con un solo nivel de profundidad.

[2] : Listado del directorio con profundidad 1. Puede observarse claramente que en los lugares donde no aparece el tipo de archivo hay un nombre de directorio, en vez de nombre de archivo, mientras que en los que si aparece el tipo, son nombres de archivo. Esto es debido a que bajo un directorio puede haber archivos con diferentes tipos cada uno y por eso no se le puede asignar un tipo determinado al directorio. Observese por ejemplo el directorio OBJECT que aparece

sin tipo, es obvio que bajo este directorio habra muchos archivos (objeto) que proceden de archivos fuente en diferentes lenguajes de programacion, por lo que no se le podra asignar un tipo especifico a este directorio. Sin embargo el archivo COBOL que es de tipo SEQDATA si es un archivo.

START

Este comando llama al compilador WFL para procesar al archivo de tipo JOB que así lo requiera; este archivo puede ser de biblioteca, guardado con anterioridad dentro de la clave, o bien el archivo de trabajo.

Es muy importante hacer notar que, como el comando START manda a que se procese un JOB, este será totalmente independiente de la sesión del usuario, desde el momento en que el sistema forma al JOB en la cola de espera que le corresponde para su ejecución. Esta es una característica sumamente útil, ya que una vez que el JOB está en cola se puede continuar con la sesión de CANDE normalmente, sin tener que esperar a ver si corrió y terminó el JOB "arrancado".

OPCIONES:

1) START

2) START <Título del Archivo>

OPCION 1:

START

Procesará el archivo de trabajo que se tenga presente en el momento de teclear al comando. El archivo DEBE SER DE TIPO JOB.

OPCION 2:

START <Título del Archivo>

Procesará el archivo especificado por su título, sin necesidad de tener presente (mediante GET) a dicho archivo, el cual DEBE SER DE TIPO JOB.

Ejemplo:

```
[1]---> MA EJEMPLO JOB
        WORKFILE EJEMPLO/ JOB
        100 BEGIN JOB EL/EJEMPLO/ES/
        200 QUEUE = 4;
        300 DISPLAY "HOLA"
        400 END JOB
[2]---> ST
[3]---> #UPDATING
        #RUNNING 7585
        JOB 7585 IN Q 4
[4]---> $
[5]---> #7585 BOJ EL/EJEMPLO/ES
        #7585 DISPLAY: HOLA.
        #7585\7585 EOJ JOB EL/EJEMPLO/ES
```

[1] : El ejemplo anterior muestra la creación de un archivo tipo JOB llamado EJEMPLO; esto se hizo a través del comando MAKE. Una vez creado el espacio de memoria para almacenar el archivo de trabajo, se procede a introducir el BEGIN JOB, QUEUE, DISPLAY y END JOB que constituyen el contenido del archivo.

[2] : Por ser este un archivo de trabajo presente, se puede dar el START sin necesidad de hacer referencia al nombre del archivo.

[3] : Inmediatamente el sistema le da un UPDATE al archivo y procede a formarlo a la cola en espera de proceso de JOBS, que en este caso fue la cuatro por ser la que se especificó dentro del JOB con la clausula QUEUE.

[4] : A partir de este momento se puede hacer de nuevo, uso normal de la terminal, e incluso despedirse mediante el BYE; esto se debe a que, a partir del momento en que el sistema indica que el JOB ya esta en la cola, este es completamente independiente de la sesion de CANDE.

[5] : Sin embargo, aqui se espero a que salieran resultados, los cuales pueden observarse en la ilustracion. La primera linea (BOJ ...), indica que el JOB ha comenzado a ejecutarse; la segunda es la ejecucion de la instruccion DISPLAY, la cual despliega la cadena de caracteres que se indico dentro del JOB. La tercera linea indica que el JOB ha finalizado su ejecucion.

El uso de la segunda opcion es similar a la anterior, con la diferencia que en la segunda se corren JOBS guardados.

?SHOW

En el momento en que se esta dentro de sesion con una terminal de la computadora, constantemente se introducen comandos de CANDE para manejar y controlar el archivo de trabajo. Estos comandos son almacenados temporalmente y procesados en una cola interna de trabajo, la cual puede ser consultada, editada y reprocesada.

El comando ?SHOW muestra precisamente esta cola de trabajo de CANDE; al utilizar este comando cada linea que se despliega, contendra un asterisco al final para indicar el fin o ultimo caracter, del comando en cuestion, y que forma parte de la cola.

Opciones:

1) ?SH

2) ?SH ALL

OPCION 1:

?SH

Muestra solamente el ultimo comando ejecutado por CANDE.

Ejemplo 1:

```
        #
(1---> F40950/A/MESP 1
        #
(2---> ?SH
(3---> F40950/A/MESP 1#
```

(1 : Aqui solo se hace uso de un comando de CANDE.

(2 : Se hace uso de ?SH.

(3 : Aparece el ultimo comando ejecutado, esto se debe al uso de la primera opcion del ?SH.

OPCION 2:

?SH ALL

Despliega por completo el contenido de la cola y cada elemento estara precedido por su numero relativo de entrada al sistema.

Los elementos no recientes seran precedidos por su numero de entrada encerrado entre parentesis cuadrados, y los elementos que acaban de ser ejecutados (recientes) tendran su numero entre parentesis triangulares; por lo general solamente el ultimo elemento de la cola tendra la caracteristica anterior, y el numero encerrado dentro del parentesis triangular sera un cero.

El comando ?SHOW no debe utilizarse cuando la terminal este ejecutando algun otro comando de CANDE o cuando este en secuencia

automatica.

En la siguiente pagina se muestra un ejemplo para ilustrar el uso del ?SHOW ALL :

Ejemplo 2:

```
(1---> HELLO MB87
       #E6700:1265 CANDE 31.280; YOU ARE DESPFII(24)
       #ENTER PASSWORD PLEASE.
       BBBBBBBBB
       # DEFAULT PRINT DESTINATION=SITE
       SEMINARIO PARA MARZO Y ABRIL
       #SESSION 0990 14:37:09 04/19/82
(2---> FILES
       (MB87) ON PACK
       . OO = DATA
       . . RPC
       . . . DATOS = DATA
       . RUN
       . . LUIS = DATA
       . . TESIS = DATA
       . SIIN
       . . OK = DATA
       . MFL
       . . RGC = JOB
       . . . MERGE = JOB
       . . . INCLUDE = JOB
       #
       G EFL/RGC/MERGE
       #NO FILE/EFL/RGC/MERGE
       ?REF.EFL.MFL
       G MFL/RGC/MERGE#
       #
       #WORKFILE MFL/RGC/MERGE: JOB, 19 RECORDS, SAVED
       L
       100 ?BEGIN JOB REBECA;
       200 ?CHARGE = MERGE;
       300 ?QUEUE = 4;
       325 ?COMPILE OBJECT/FORTRAN/REBECA FORTRAN;
       350 ?FORTRAN FILE CARD(DISK,FILETYPE=7);
       #
       F300.4.5
       #
       L=
       300 ?QUEUE = 5;
       #
(3---> ?SH A
       (4)FILES#
       (3)G MFL/RGC/MERGE#
       (2)L#
       (1)F300.4.5#
```

[1] : Se entra en sesion.

[2] : Se ejecutan varios comandos.

[3] : Se ejecuta el comando ?SH ALL y la maquina lista todos los comandos que se han ejecutado en la sesion.

Se puede observar que el comando GET no fue ejecutado la primera vez, debido a que se especifico un nombre de archivo que no existe en la biblioteca de la clave. Para corregir esto se utiliza el comando ?REP y se corrige el nombre del archivo, con lo cual ahora si se efectua la accion del comando y se guarda en cola.

Lo explicado en el parrafo anterior puede verificarse con el mismo ?SH A; como se puede ver, en el desplegado de la cola de comandos ejecutados solo aparece el segundo GET, ya que el primer GET fue corregido y repetido mediante el comando ?REP. Tambien hay que mencionar que los comandos de control, tales como el ?REP en este caso, no pasan a formar parte de la cola de comandos de CANDE, tal y como puede observarse en el mismo ejemplo.

?RETRIEVE

El comando ?RETRIEVE extrae un elemento de la cola de comandos, el cual se le indica por medio del numero de entrada correspondiente, y lo reacomoda en el sitio mas reciente de la cola, como si hubiera sido el ultimo comando ejecutado, aunque no se ejecuta. El comando en cuestion solo se cambia de lugar dentro de la cola. El numero de comando indicado debe ser mayor a cero y menor o igual al numero maximo de elementos en cola; si este numero no es especificado entonces el sistema asumira el numero 1.

OPCIONES :

1) ?RET <#Elem>

2) ?RET <#Elem> REPEAT <Especificaciones de Edicion>

OPCION 1:

?RET <#Elem>

Su accion es la indicada en el parrafo anterior.

EJEMPLO:

```
?SH A
[5]G WFL/RGC/MERGE$
[4]$
[3]F300.4.5$
[2]=$
[1]UPDATE$
<0>FEN$
#
?FET 5
#
?SH A
[5]$
[4]F300.4.5$
[3]=$
[2]UPDATE$
[1]REN$
<0>G WFL/RGC/MERGE$
#
```

En este ejemplo se puede ver, mediante el ?SH A, el contenido de la cola de comandos ejecutados de CANDE. Con el comando ?RET se esta reacomodando el numero 5 de la cola para colocarlo en la posicion de comandos mas recientes.

OPCION 2:

?RET <#Elem> REPEAT <Especificaciones de edicion>

Con esta opcion si se ejecuta la instruccion indicada por el comando que se mando recobrar con el ?RET; las especificaciones de edicion se refieren a la posibilidad de hacer modificaciones al texto que se mando recobrar y repetir, tal y como si se le estuviera haciendo un FIX en su forma mas sencilla. El ?REP produce el despliege del texto antes de ser ejecutado.

EJEMPLO:

```
?SH A
[5]F300.4.5$
[4]=$
[3]UPDATE$
[2]REN$
[1]G WFL/RGC/MERGE$
<0>FILE WFL$
#
?RET REP/MERGE/INCLUDE
G WFL/RGC/INCLUDE$
#
#UOPENFILE WFL/RGC/INCLUDE:JOB,20 RECORDS, SAVED
```

Este ejemplo muestra el uso del ?RET combinado con la opcion REP, la cual produce la ejecucion del comando recobrado, que en este caso fue el numero 1. Ademas se utilizan especificaciones de edicion, que aqui sirven para cambiar el nombre del programa obtenido mediante el GET y que ahora ya no sera WFL/RGC/MERGE sino WFL/RGC/INCLUDE.

Se observa que el elemento asumido por el comando ?RET fue el numero uno pues es el que toma por omision cuando no se especifica el numero del elemento que se desea recobrar.

El comando ?RET no debe utilizarse cuando la terminal este procesando algun comando de cande o cuando este en secuencia automatica.

El ?RET combinado con REP es muy util cuando por ejemplo se tienen que utilizar repetidamente comandos de cande cuyo texto es demasiado largo, y por lo tanto engorrosos y tardados si se introducen directamente.

Un ejemplo clasico de esta situacion son las corridas de programas con extensas ecuaciones de archivo.

?REPEAT

Este comando ordena a CANDE que ejecute el ultimo elemento introducido en la cola de comandos ya ejecutados, tal y como si se le estuviese introduciendo el comando a traves del teclado.

OPCIONES :

1) ?REPEAT (Especificaciones de Edicion)

Las especificaciones de edicion son opcionales, sirven para modificar el texto repetido el cual sera desplegado antes de ser ejecutado. Este comando no debe ser utilizado cuando la terminal este procesando otros comandos o en secuencia automatica.

Ejemplo:

```
?SH
SAT
#
?REP.SA.SA AS WFL/RGC/MEZCLA
#
SA AS WFL/RGC/MEZCLAT
#UPDATING
#WORKSOURCE WFL/RGC/MERGE SAVED AS (MB87)WFL/RGC/MEZCLA ON PI
```

Se da el comando ?SH para ver cual fue la ultima instruccion de CANDE que fue ejecutada, viendo que fue un SAVE, entonces se quiere que el sistema repita la instruccion, pero que ahora guarde el programa con el nombre indicado por el cambio. El comando repetido y editado se despliega y a continuacion se ejecuta.

Se puede observar que las especificaciones de edicion se trabajan de manera similar a un FIX en su forma mas sencilla, se introduce el comando ?REP y a continuacion se le dice que donde

se encuentra: SA (dentro del elemento(comando) en cuestion), que ahora se escriba : MFL/RGC/MEZCLA.

SECURITY

Esta instruccion permite modificar la seguridad que tenga un archivo en disco, es decir la disponibilidad de uso que de ese archivo puedan tener otros usuarios no poseedores de la clave bajo la cual se encuentra dicho archivo.

Los tipos de seguridad son:

1) PRIVATE: Solo permite el uso del archivo al dueno de la clave o claves privilegiadas.

2) PUBLIC ID: Cualquier usuario bajo cualquier clave puede listar, ejecutar y modificar el archivo en cuestion.

3) PUBLIC SECURED: El archivo no puede ser listado ni modificado, unicamente puede ser ejecutado (si el objeto existe).

4) PUBLIC IN: El archivo solo puede ser leido.

5) PUBLIC OUT: Unicamente puede escribirse sobre ese archivo.

OPCIONES :

1) SEC SOURCE (Nombre del Archivo) (tipo de seguridad)

2) SEC OBJECT (Nombre del Archivo) (tipo de seguridad)

OPCION 1:

SEC SOURCE (Nombre del Archivo) (tipo de seguridad)

Nos permite modificar la seguridad del archivo fuente.

OPCION 2:

SEC OBJECT <Nombre del Archivo> <tipo de seguridad>

Nos permite modificar la seguridad del archivo objeto.

Ejemplos:

```
[1]---> LFILES EJEMPLO:SECURITY
        #RUNNING 8377
[2]---> (JPS1) - DIFECTORY ON PACK
        . EJEMPLO : FORTRANSYMBOL SECURITY=PRIVATE (I/O)
        #
[3]---> SEC EJEMPLO PUBLIC IO
        # (JPS1)EJEMPLO ON PACK SECURITY CHANGED
        LFILES EJEMPLO:SECURITY
        #RUNNING 8378
[4]---> (JPS1) :DIFECTORY ON PACK
        . EJEMPLO :FORTRANSYMBOL SECURYTY=PUBLIC (I/O)
        #
[5]---> SEC EJEMPLO PRIVATE
        # (JPS1)EJEMPLO ON PACK SECURITY CHANGED
```

[1] : El comando LFILES<Nombre del Archivo>:SECURITY se usa para conocer el tipo de seguridad que tiene el archivo al que nos referimos.

[2] : Para este ejemplo se dice que la seguridad de EJEMPLO es de tipo PRIVATE (es decir que unicamente el poseedor de la clave tiene acceso a el).

[3] : Por medio de la instruccion SEC EJEMPLO PUBLIC IO la seguridad de EJEMPLO fue cambiada como efectivamente se ve en la respuesta de la maquina.

[4] : Aqui se vuelve a hacer uso del comando LFILES para preguntar la seguridad del archivo EJEMPLO. En este caso responde que la seguridad de dicho archivo ha sido cambiada a Public (I/O)

[5] : Finalmente a traves del comando SEC se le devuelve al archivo EJEMPLO la seguridad de tipo PRIVATE.

+ Por default al crear un archivo y salvarlo se le da seguridad privada.

?SS

El comando ?SS manda un mensaje a otra terminal o al operador al cual se le identifica de la siguiente manera: SPO

Ejemplo: se le manda el siguiente mensaje al operador

```
?SS SPO DISCULPE SE ENCUENTRA CARGADO EL PACK UNAM2
```

La manera en que se recibe la respuesta es:

```
#14:05 FROM SPO:SI
```

Si se desea comunicarse con otra persona que se encuentre en otra division o terminal se debe hacer lo siguiente:

a) Saber a quien se le mandara el mensaje, para esto basta conocer la clave del usuario con el que nos queremos comunicar.

b) Saber el lugar donde esta trabajando y el numero logico de la estacion (ver comando ?WHERE).

Ejemplo:

```
?SS 70 ESCRIBE ALGO ILUSTRATIVO PARA EL MANUAL DE CANDE
```

El primer numero es el de la estacion , lo que sigue es el mensaje mandado.

La forma de recibir los mensajes es la siguiente:

```
#13:56 FROM MCBI ON 70: ESTO ES UN MENSAJE !
```


Como se observa nos despliega la hora, la clave de la persona que manda el mensaje y el numero logico de la estacion, es importante tomar en cuenta este ultimo numero por si se desea contestar al remitente.

?WRU

El comando ?WRU despliega la identidad de la estacion del usuario, asi como la sesion en la que se encuentra y su usercode.

Ejemplo:

```
?WRU
#66700:1268 CANDE 31.280: YOU ARE FCIENCMAT3(35)
#SESSION = 4097 USER = MCBI
```

Tambien como se ve en el ejemplo, proporciona otros datos referentes al sistema.

Como se sabe, todos los comandos de control, pueden ser usados antes de entrar a sesion, en este caso si lo hacemos de esta manera la respuesta es la siguiente:

```
?WRU
#66700:1268 CANDE 51.280: YOU ARE FCIENCMAT3(35)
```

La sesion y el usercode no aparecen, puesto que no se ha entrado a sesion.

?WHERE

El comando ?WHERE despliega el nombre de la(s) estacion(es) donde hay alguien trabajando con la clave indicada, asi como el numero de la estacion donde se encuentra(n).

Ejemplo:

```
?WHERE MCBI
# MCBI ON DESPFII(24)
# MCBI ON DIMEF103(70)
# MCBI ON FCIENCHAT(11)
```

Lo que se hizo fue preguntar donde se encuentra trabajando la clave dada, en este caso MCBI, inmediatamente se obtuvo la respuesta, por ejemplo:

```
?WHERE FIIBI
# FIIBI ON FCIENCHAT3(35)
# FIIBI ON USUACUD4(56)
```

Si ninguna persona la esta usando, nos aparece:

```
?WHERE JL89
# JL89 NOT ON
?WHERE AMSU
# AMSU NOT ON
```


?WD

El comando ?WD despliega el dia corriente de la siguiente forma: dia de la semana, mes, dia del mes, ano y hora.

Ejemplo:

```
?WD
DATE IS MONDAY MAY 03,1982 (82123) 15:07:14
```

El numero encerrado entre parentesis da el ano vigente y los dias transcurridos.

?WT

El comando ?WT despliega el dia de la semana, el mes, dia del mes, ano y hora en minutos y segundos.

Ejemplo:

```
?WT
DATE IS MONDAY MAY 03,1982 (82123) 15:07:03
```

Como se observa este comando realiza lo mismo que el comando ?WD.

?TIME

Existe otro comando que realiza lo mismo que los dos comandos anteriores y es precisamente el comando ?TIME.

Ejemplo:

```
?TIME
#3:07 PM MONDAY , MAY 3 , 1982
```

La diferencia es obvia: el orden es diferente, la hora es dada en modulo 12, ademas no da los segundos.

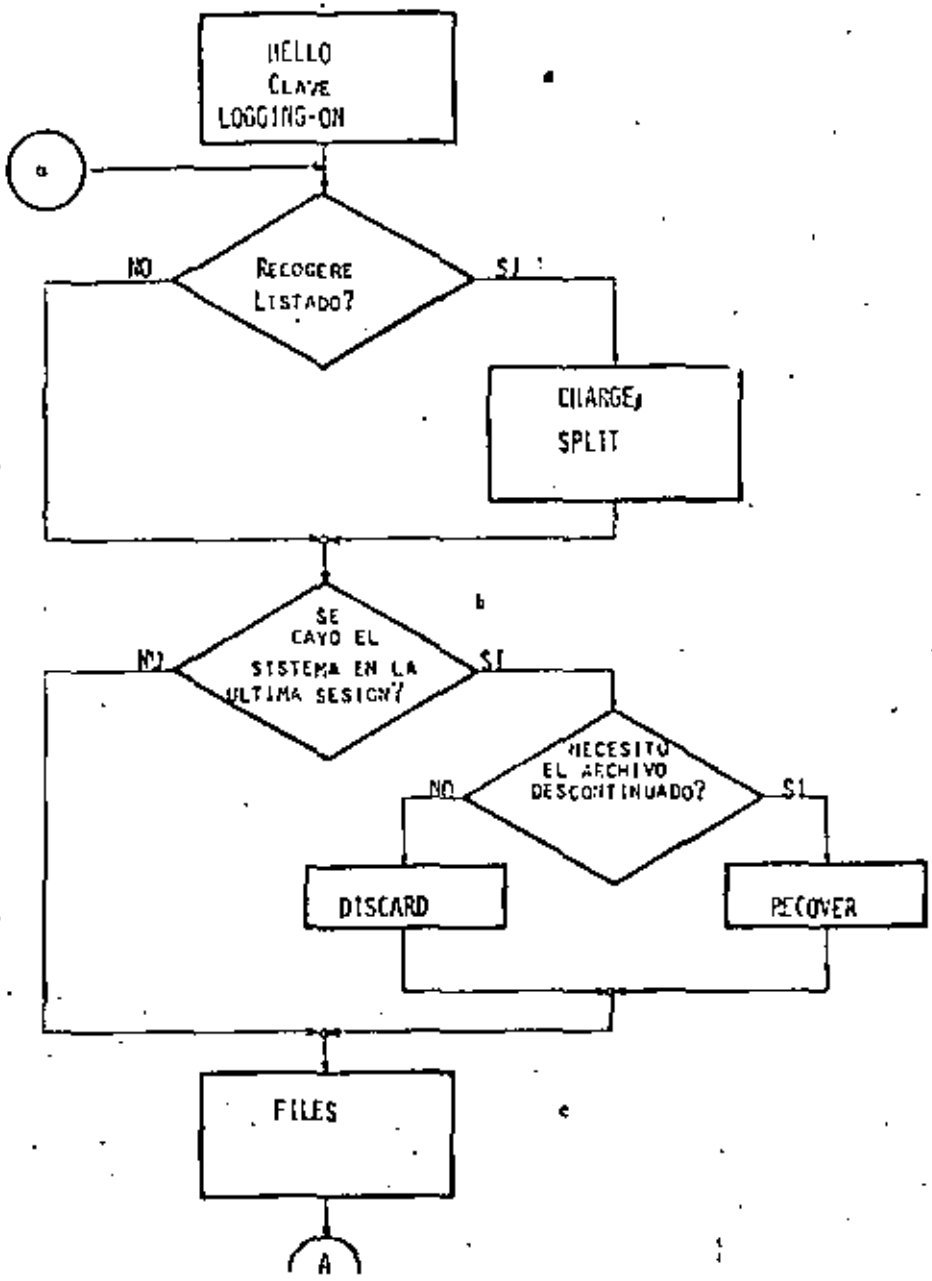
En los siguientes ejemplos se ve la diferencia:

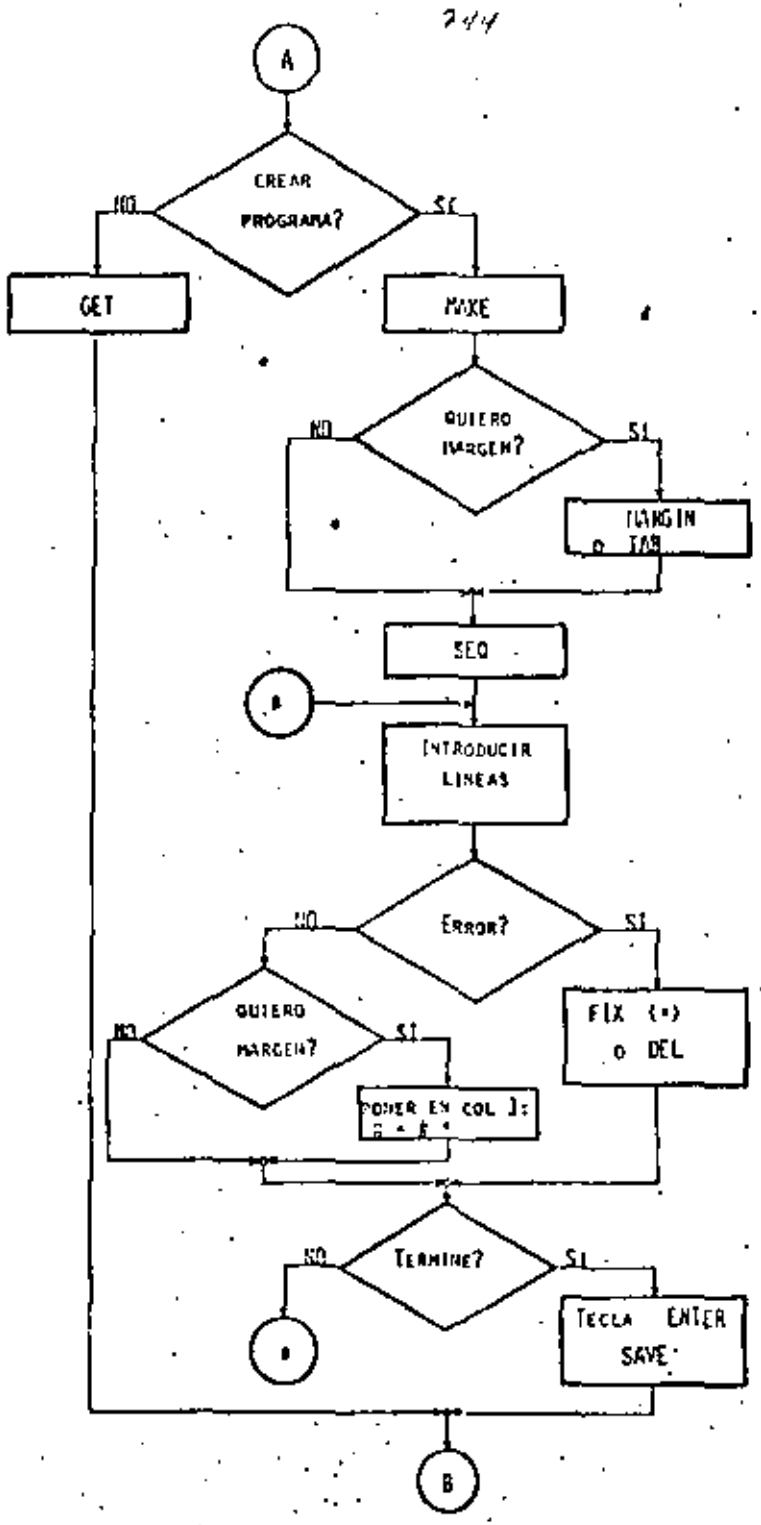
```
?WD
DATE IS MONDAY MAY 03,1982 (82123) 15:06:46
?WT
DATE IS MONDAY MAY 03,1982 (82123) 15:06:57

?TIME
#3:07 PM MONDAY , MAY 3 , 1982
```

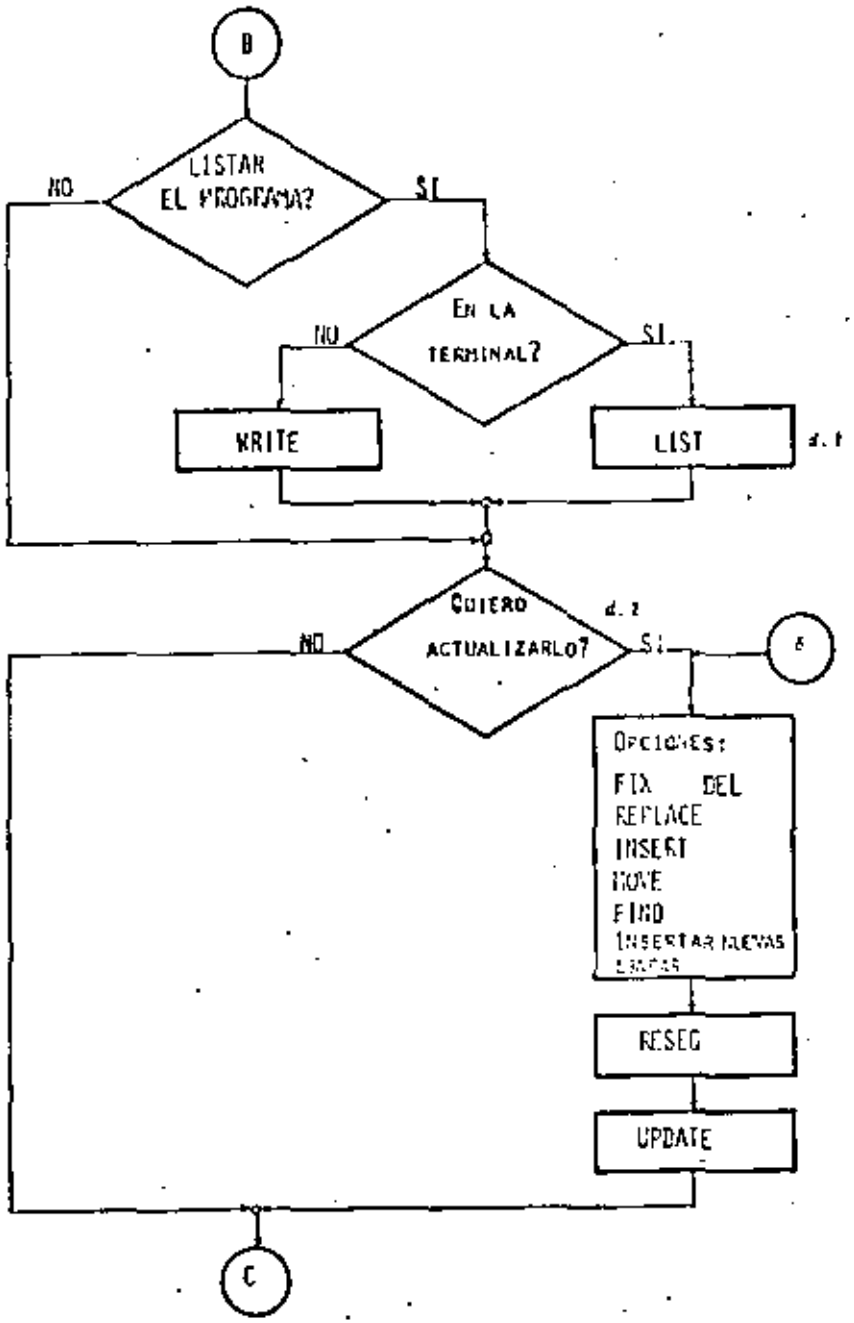

BIBLIOGRAFIA

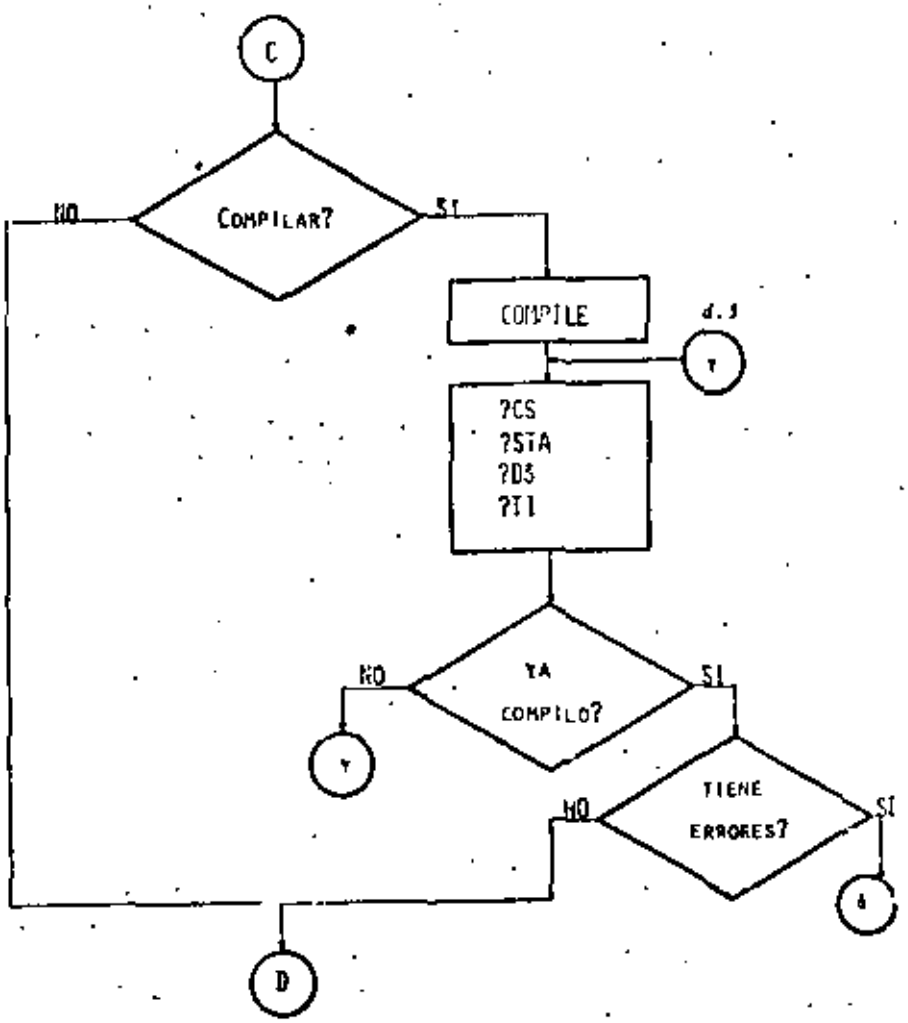
BURROUGHS B6700/B7700.
CANDE
Reference Manual
Burroughs Corporation
1978

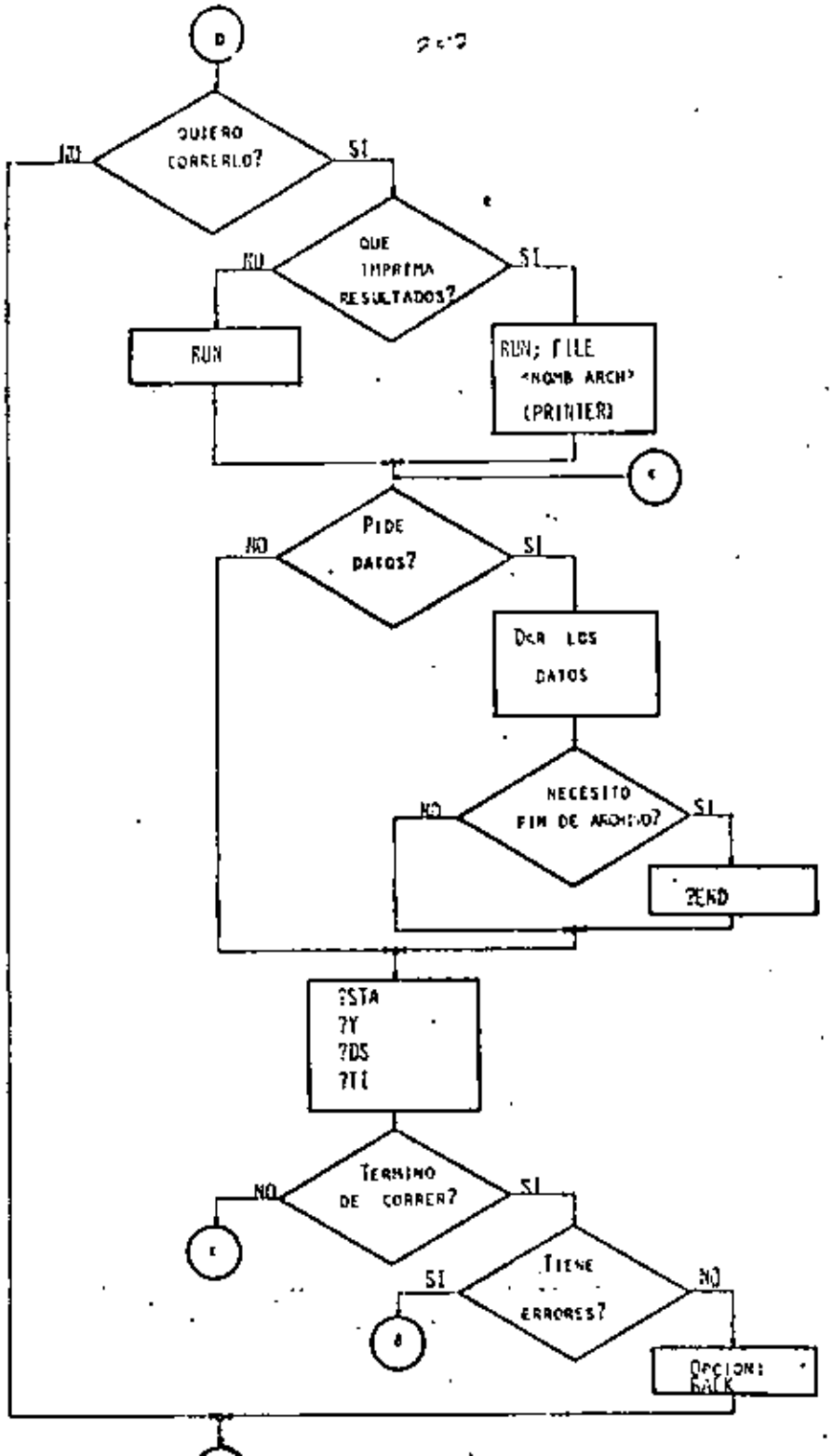
Diagrama de las opciones del editor CANDE

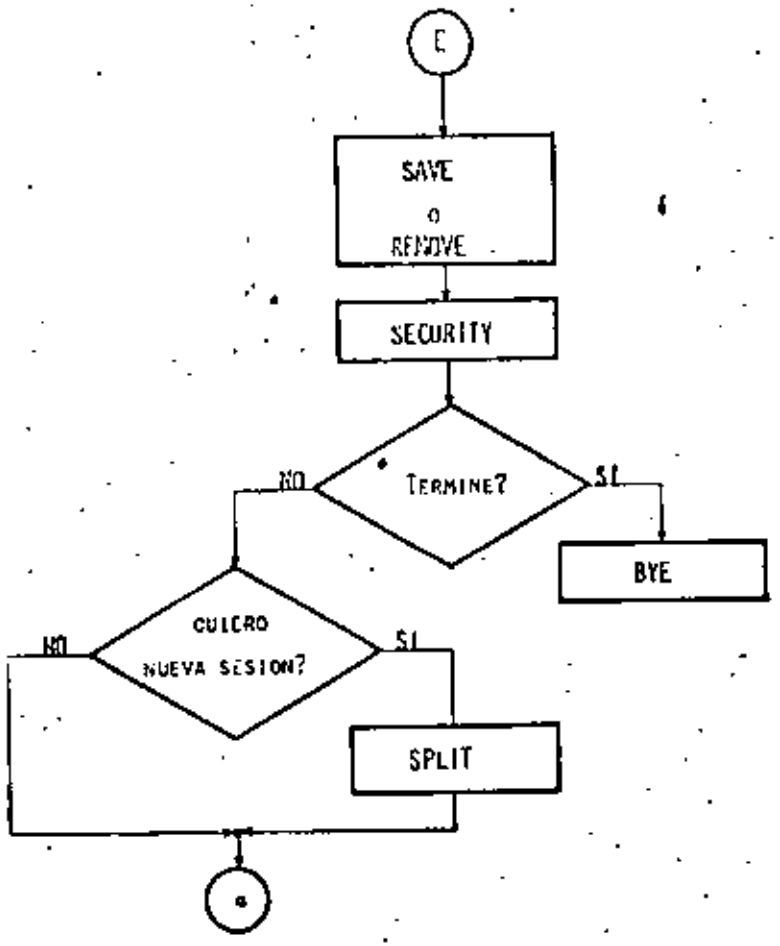


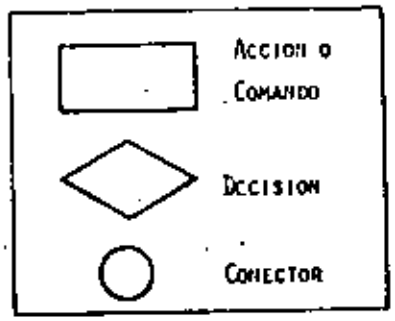


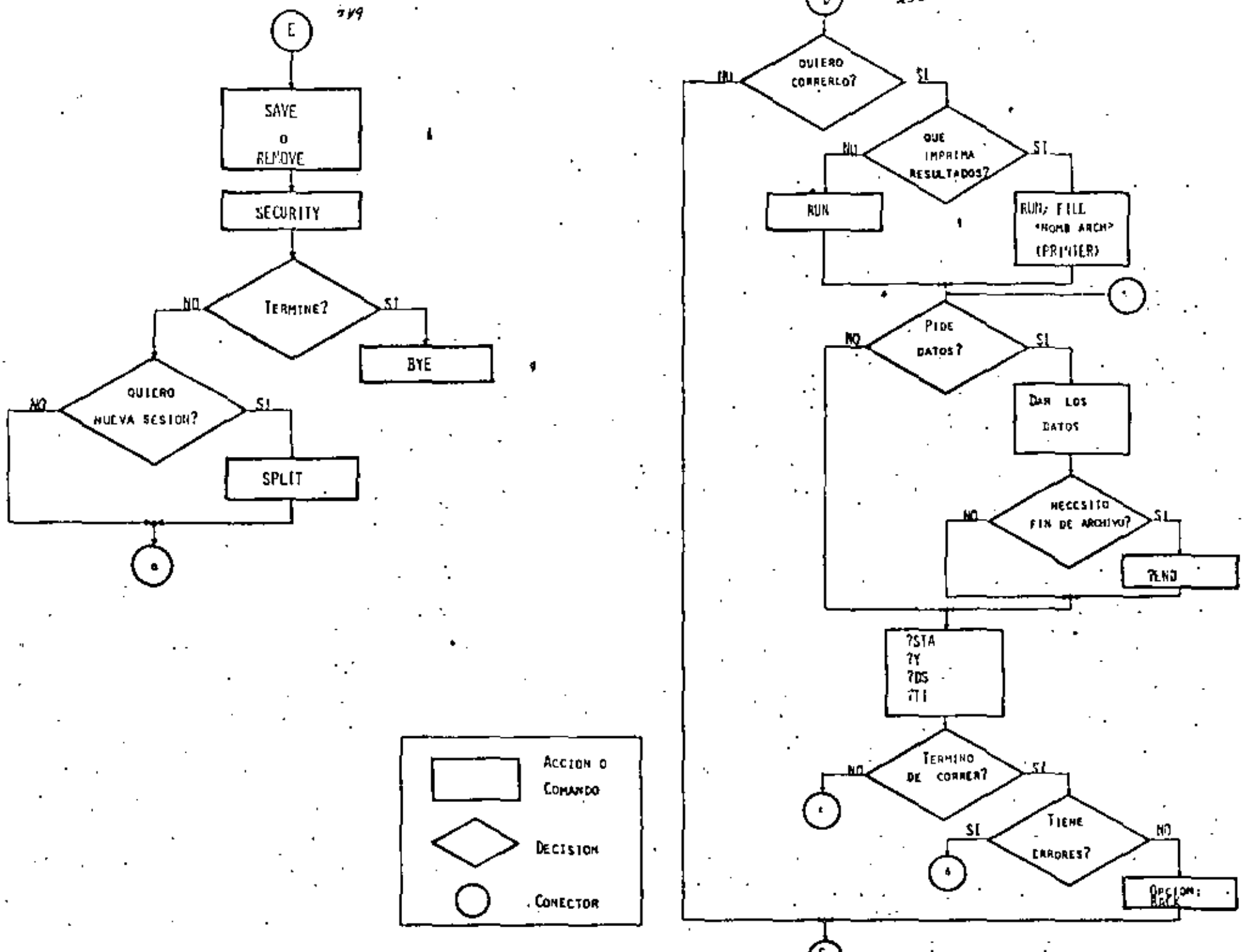

B
LISTAR EL PROGRAMA?
NO
SI
EN LA TERMINAL?
NO
SI
WRITE
LIST
d.1
QUIERO ACTUALIZARLO?
d.2
NO
SI
6
OPCIONES:
FIX DEL
REPLACE
INSERT
MOVE
FIND
INSERTAR NUEVAS LINEAS
RESEQ
UPDATE
C

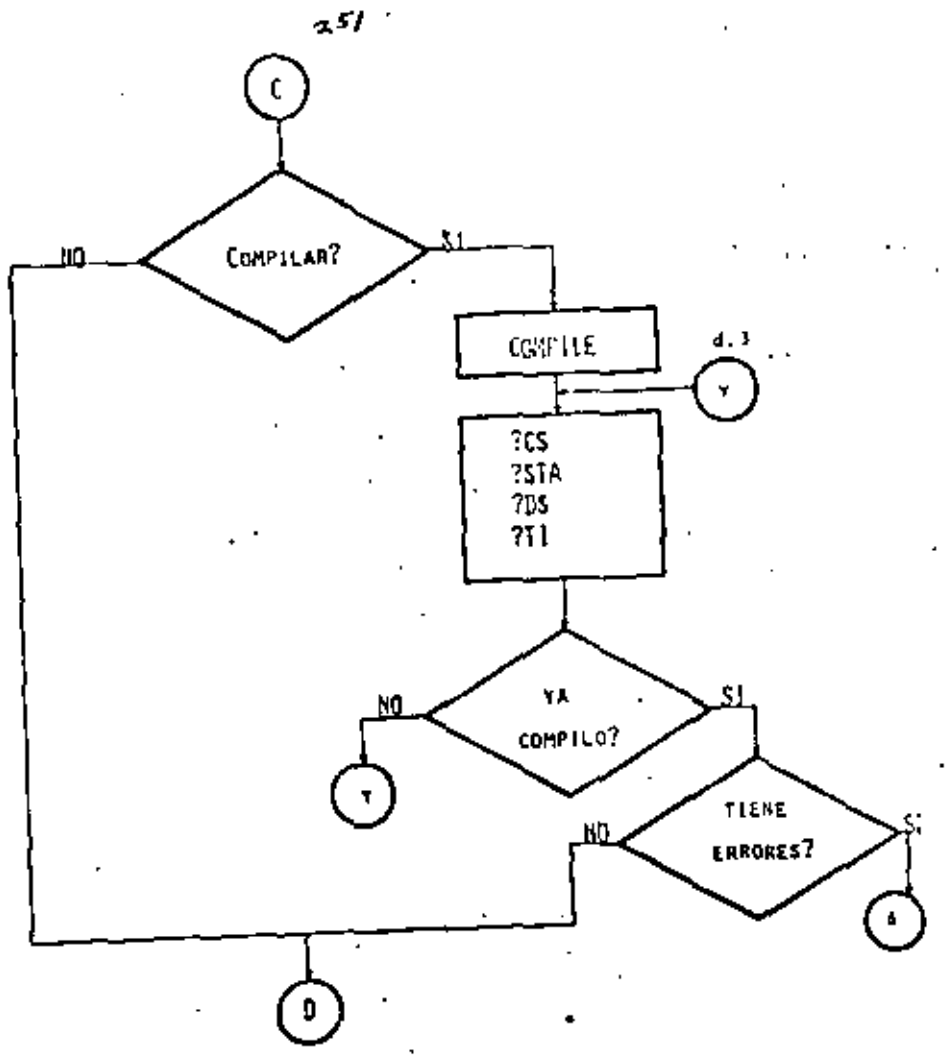

C
COMPILAR?
NO
SI
COMPILE
d.3
7
?CS
?STA
?DS
?IT
YA COMPILO?
NO
SI
7
TIENE ERRORES?
NO
SI
6
D


D
QUIERO CORRERLO?
NO
SI
QUE IMPRIMA RESULTADOS?
NO
SI
RUN
RUN; FILE <NOMB ARCH> (PRINTER)
C
PIDE DATOS?
NO
SI
DAR LOS DATOS
NECESITO FIN DE ARCHIVO?
NO
SI
?END
?STA
?Y
?DS
?TI
TERMINO DE CORRER?
NO
SI
C
TIENE ERRORES?
SI
NO
B
OPCION: BACK


E
SAVE
O
REMOVE
SECURITY
TERMINE?
NO
SI
BYE
QUIERO NUEVA SESION?
NO
SI
SPLIT
B


ACCION O COMANDO
DECISION
CONECTOR


E
SAVE
o
REMOVE
SECURITY
TERMINE?
NO
SI
BYE
QUIERO
NUEVA SESION?
NO
SI
SPLIT
D
QUIERO
CORRERLO?
NO
SI
QUE
IMPRIMA
RESULTADOS?
NO
SI
RUN
RUN; FILE
<NOMB ARCH>
(PRINTER)
PIDE
DATOS?
NO
SI
DAR LOS
DATOS
NECESITO
FIN DE ARCHIVO?
NO
SI
?END
?STA
?Y
?DS
?T
TERMINO
DE CORRER?
NO
SI
TIENE
ERRORES?
SI
NO
OPCION:
ACCION O
COMANDO
DECISION
CONECTOR


251
C
COMPILAR?
NO
SI
COMPILE
d.3
Y
?CS
?STA
?DS
?T1
YA COMPILO?
NO
SI
Y
TIENE ERRORES?
NO
SI
A
D

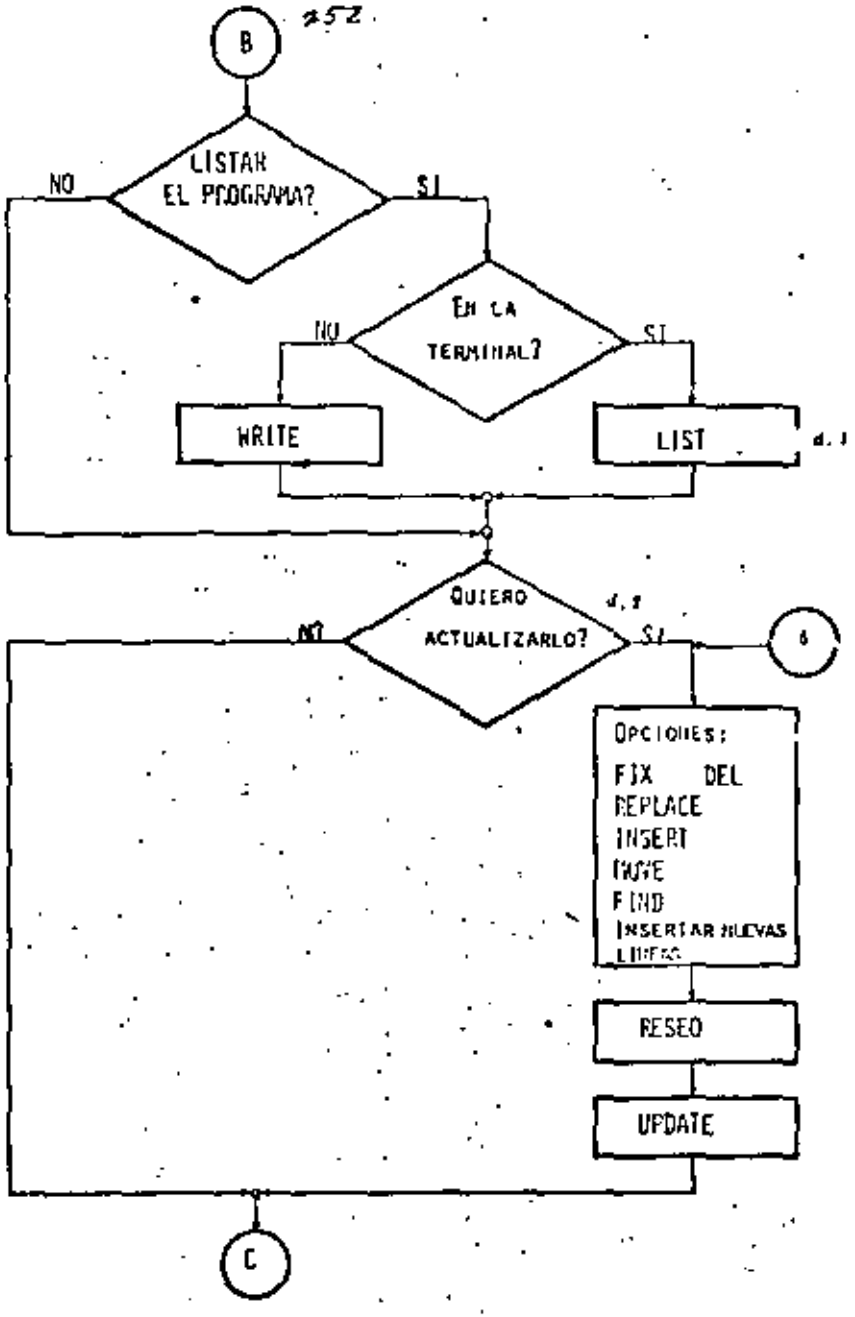
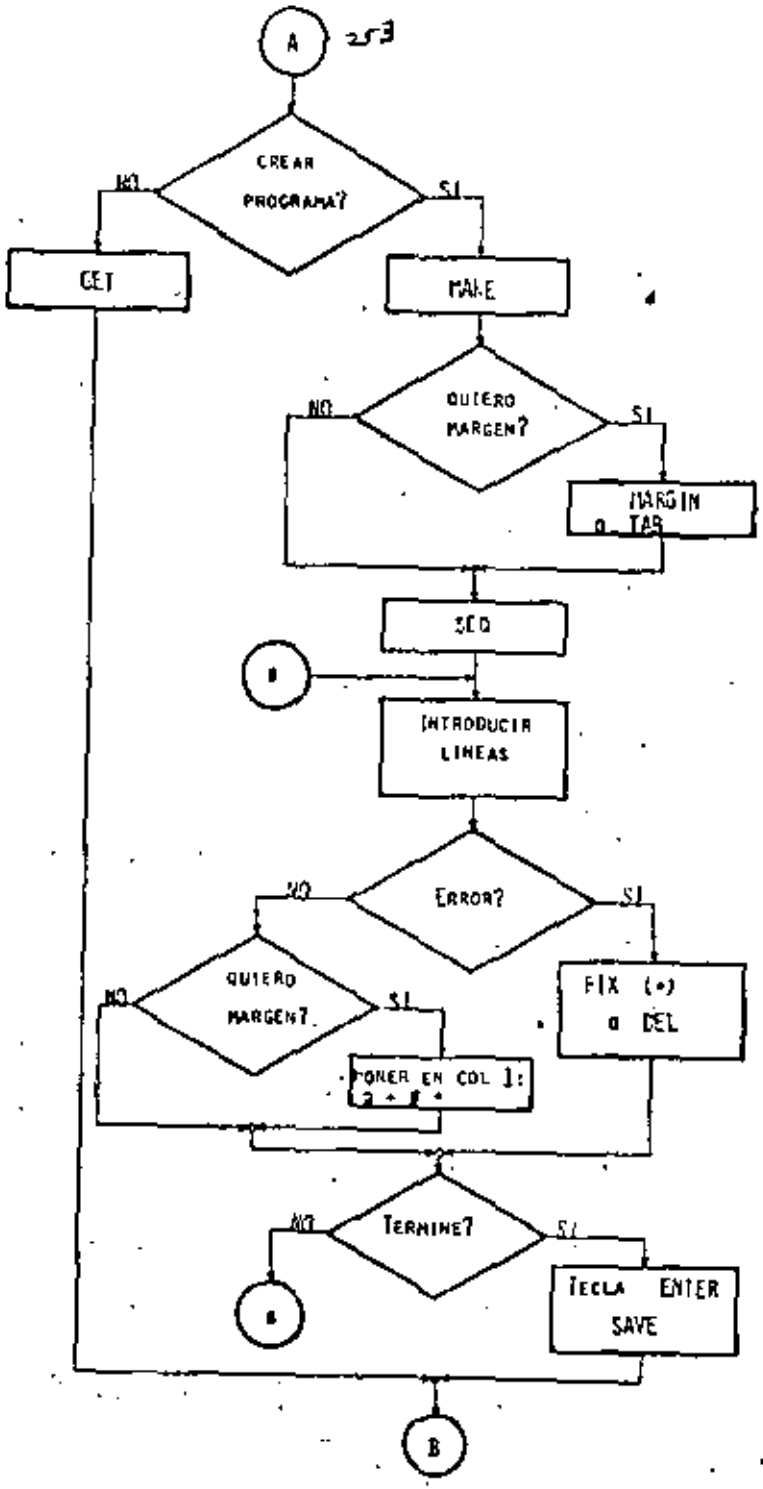
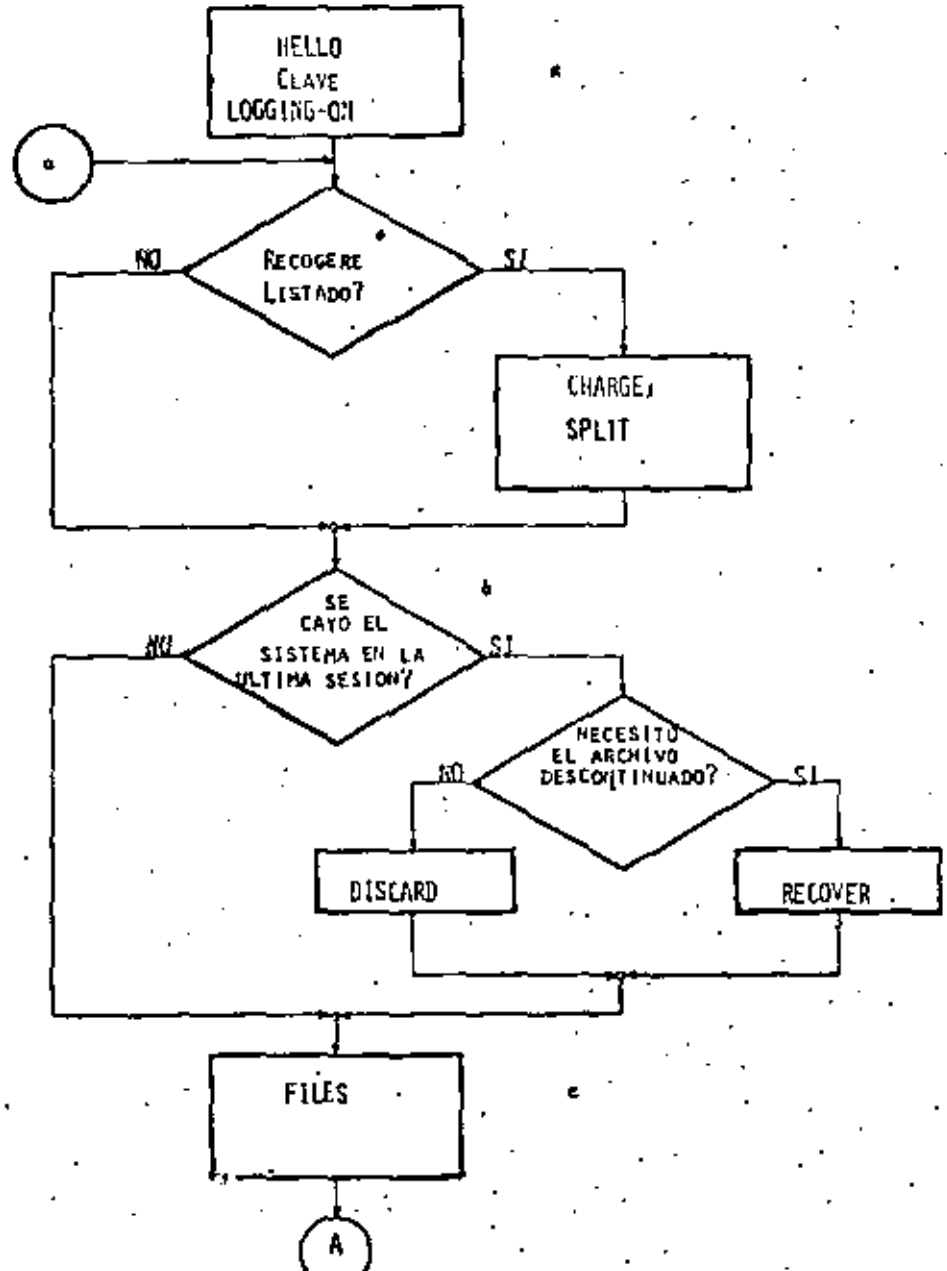

252
B
LISTAR EL PROGRAMA?
NO
SI
EN LA TERMINAL?
NO
SI
WRITE
LIST
d.1
QUIERO ACTUALIZARLO?
d.2
NO
SI
A
OPCIONES:
FIX DEL
REPLACE
INSERT
MOVE
FIND
INSERTAR NUEVAS LINEAS
RESEQ
UPDATE
C

Diagrama de las opciones del editor CANDE





DIRECTORIO DE ASISTENTES AL CURSO CANDE (Comandos y Edición) DEL 11 AL 17 DE MAYO DE 1983

NOMBRE Y DIRECCION	EMPRESA Y DIRECCION
1. FERNANDO CASTREJON SERRANO Privada de Arteaga No. 119 Altos 1 Centro Cuernavaca, Morelos 2 65 80	UNIVERSIDAD AUTONOMA DEL ESTADO DE MORELOS Av. Universidad 1001 Chamilpa Cuernavaca, Morelos 3 26 44
2. FRANCISCO CORTEZ CAMACHO Venturina 116 Col. Estrella Deleg. G. A. Madero C.P. 07810 México, D. F. 5 77 24 55	ANALISIS Y PROYECTOS DE INGENIERIA,S.A. de C. V. Mitla No. 324 Col. Narvarte Deleg. Benito Juárez C.P. 03020 México, D. F. 6 96 51 77
3. MARISELA ESTRADA GARCIA Av. Río No. 111 Casa 1517 Col. Chimalistac Deleg. Alvaro Obregón C.P. 01070 México, D. F. 5 50 82 58	DIRECCION GENERAL DE OBRAS Av. Revolución 2045 Deleg. Alvaro Obregón C.P. 01070 México, D. F. 5 50 57 83
4. LUIS GABRIEL FERNANDEZ SANCHEZ Av. Ermita Izt. No. 499 Col. Esmeralda Deleg. Iztapalapa C.P. 09810 México, D. F. 5 82 00 29	
5. J. JESUS GARCIA ESPINOSA Sur 111 "A" No. 711 Sector Popular Deleg. Iztapalapa C.P. 09060 México, D. F. 5 82 76 84	DIRECCION GENERAL DE OBRAS Av. Revolución 2045 Deleg. Alvaro Obregón C.P. 01070 México, D. F. 5 50 57 83
6. JOSE DE JESUS GARCIA HERNANDEZ José Ceballos 43-2 San Miguel Chapultepec Deleg. Miguel Hidalgo México, D. F. 5 16 57 58	PROGRAMA UNIVERSITARIO DE COMPUTO Ciudad Universitaria México, D. F.