FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

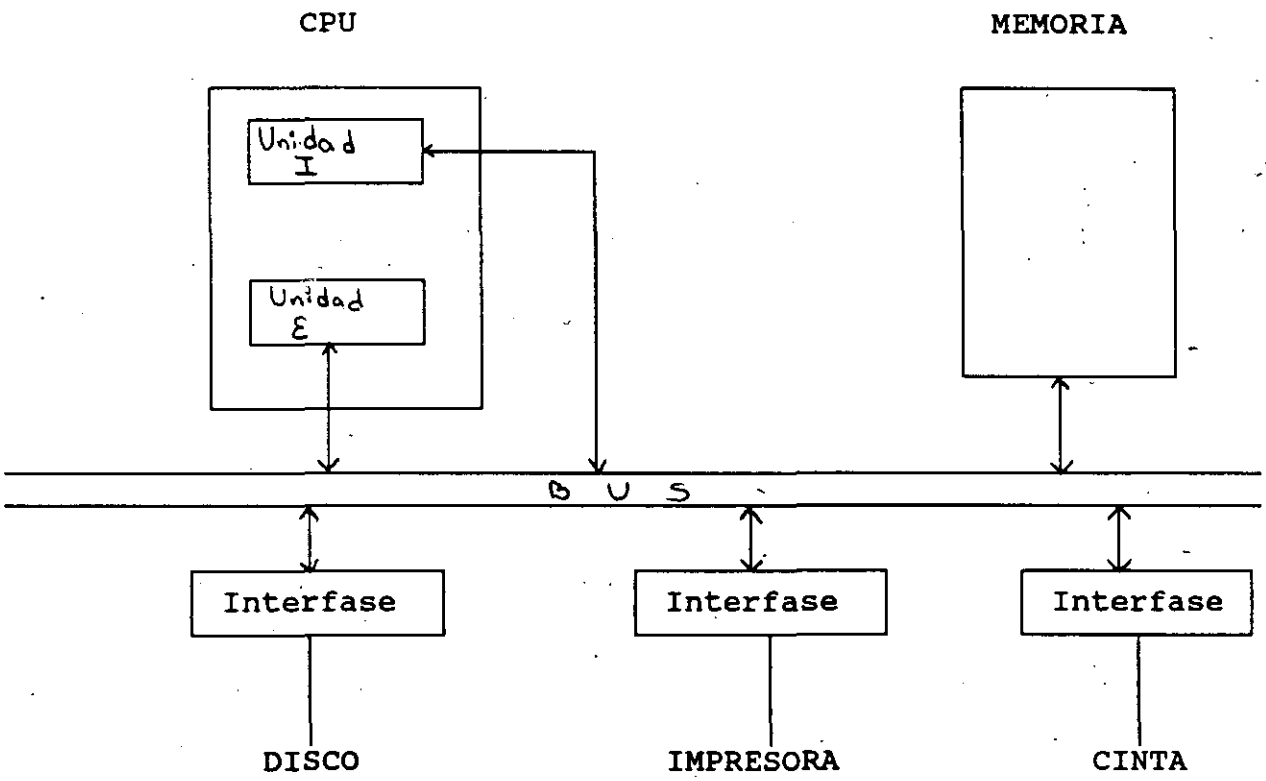
SISTEMAS OPERATIVOS Y SUS FUNCIONES

MATERIAL DIDACTICO

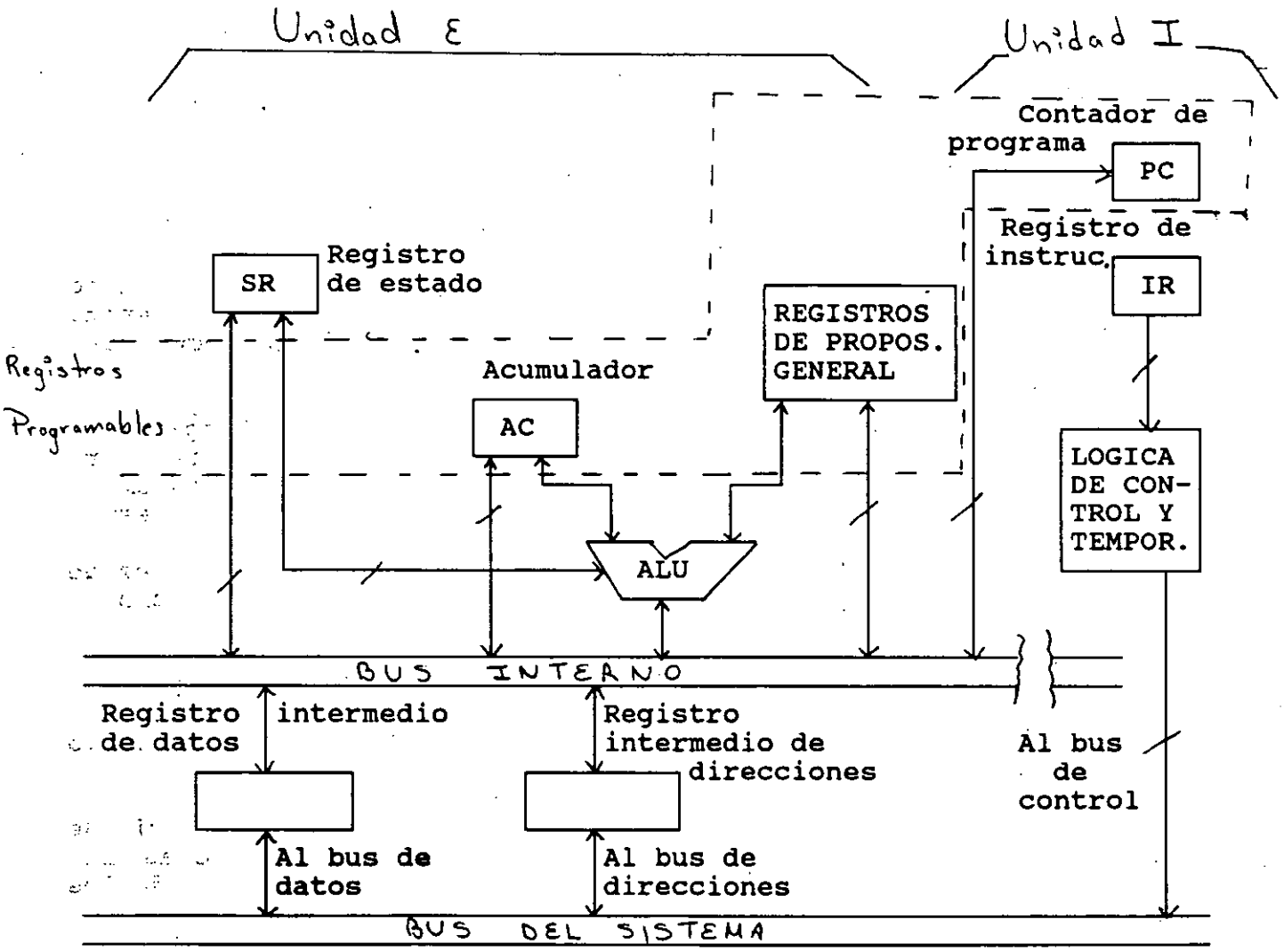
DICIEMBRE, 1992.

ESTRUCTURA Y ORGANIZACION DE LOS EQUIPOS DE COMPUTO

Organización de un sistema básico de computación.



Organización Interna de un procesador.



SISTEMA OPERATIVO

Un sistema operativo es un programa que siempre esta en ejecución, el cual se puede ver como una colección organizada de software (módulos) que entiende el hardware y que consta de rutinas de control para la operación y administración de los recursos de una computadora.

El sistema operativo es una interfaz con:

*) El hardware
*) Los programadores de aplicaciones
*) Los programadores de sistemas
*) Los operadores del computador
*) Los programas

Los programas se acoplan con los sistemas operativos por medio de instrucciones especiales, dentro de las cuales se tienen: Llamadas al supervisor, llamadas al monitor, peticiones ejecutivas, etc.

Los operadores del computador: Son las personas encargadas de la vigilancia del sistema operativo, respondiendo a peticiones para intervenir, montando y desmontando discos o cintas, cargando o descargando tarjetas, asegurándose de que las impresoras estén cargadas con los formatos adecuados y que estos formatos estén bien alineados, etc. Los operadores son parte integral de la fluidez de operación de un sistema; son quienes desempeñan aquellas funciones que aún no se han automatizado.

Los programadores del sistema: Están relacionados, generalmente, con el mantenimiento del sistema operativo, adaptándolo a las necesidades de la instalación y modificándolo para poder soportar nuevos tipos de dispositivos.

Al sistema operativo se le da, por lo general, la categoría de usuario de mayor confianza. Se le permite acceder a todas las características del hardware, a todos los programas y datos de los usuarios, etc.

Los recursos sobre los que actúa un sistema operativo son:
*)Memoria
*)Dispositivos de E/S
*)El (Los) procesador (es)
*)Información (archivos)

CONCEPTOS BASICOS

Trabajo:Es un programa en formato ejecutable que esta por ejecutarse.

Proceso o tareas:En términos generales, proceso es la "manipulación" de la información, es decir, es cualquier actividad que se realiza con la información en el interior de la computadora, la cual puede modificar el contenido o la forma de la información o sólo la transfiere de una fuente a otra sin alterarla.

Estado de ejecución:Es cuando un proceso se ha inicializado en el computo y aún no se ha terminado.

Básicamente existen dos estados de ejecución:

a)El estado esclavo o estado usuario o estado problema, en el cual el procesador ejecuta las instrucciones de los programas de usuarios.

b)El estado amo o estado supervisor en el cual el procesador puede ejecutar correctamente las instrucciones privilegiadas.

Instrucciones privilegiadas:Son instrucciones del sistema que generalmente no están disponibles para el usuario, sólo son ejecutables por el sistema operativo.

Ejemplos de instrucciones privilegiadas:
*)Inicializar los procesadores de E/S
*)Cambiar los derechos de protección de memoria
*)Cambiar el estado de interrupción de la máquina

Interrupción:Es un mecanismo por el cual se forza a un procesador a "dejar" (suspender) la tarea que está realizando para atender a otra.
Generaciones de los sistemas operativos.

Generaciones de computadoras

Los sistemas operativos, al igual que el hardware de los computadores, han sufrido una serie de cambios revolucionarios llamados generaciones. En el caso del hardware se tienen las siguientes generaciones:

Primera generación:Uso de válvulas al vacío (1945-1955)
Segunda generación:Uso del transistor (1955-1965)
Tercera generación:Uso de circuitos integrados (1965-1980)
Cuarta generación:Circuitos integrados a gran escala (1980-)

Cada generación tuvo un aumento notable en cuanto a:
-Capacidad
-Rapidez
-Precisión
-Velocidad

Y además cada generación ha ido acompañada de reducciones sustanciales en cuanto a:
-Costos
-Tamaño
-Emisión de calor
-Consumo de energía

GENERACIONES DE LOS SISTEMAS OPERATIVOS

Generación Cero de los sistemas operativos "Sistemas de control de E/S". (Década de 1940)

Los sistemas computacionales de la primera generación no poseían sistema operativo. La programación y ejecución de los programas se realizaban en la máquina "desnuda", es decir, los usuarios tenían completo acceso al lenguaje de máquina. Todas las instrucciones y los datos eran codificados a mano en código binario.

Primera generación de los sistemas operativos (Década de 1950)

Los sistemas operativos de esta generación fueron diseñados para hacer más fluida la transmisión entre trabajos. Antes de que los sistemas fueran diseñados, se perdía un tiempo considerable entre la terminación de un trabajo y el inicio del siguiente.Por lo que la característica de esta generación es la aparición del procesamiento por lotes (Batch) y por consiguiente los sistemas operativos por lotes, donde los trabajos se reunían por grupos o lotes.

Sistemas operativos por lotes.

Son sistemas operativos en los cuales las instrucciones, los datos y las órdenes del sistema están reunidos en forma de trabajo. Este tipo de sistemas operativos permiten poca o ninguna iteración entre el usuario y los programas en ejecución.

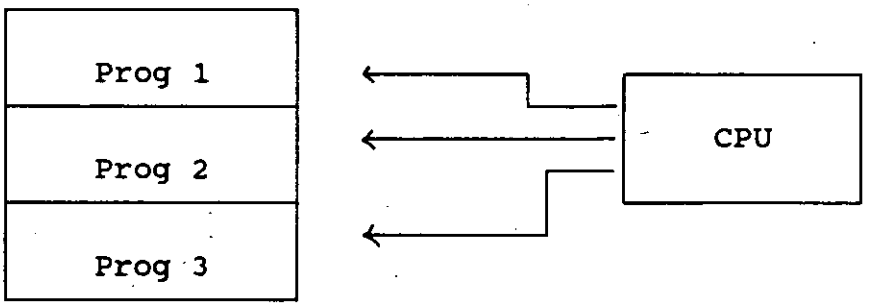
Segunda generación de los sistemas operativos (Primer mitad de la década de 1960)

Con las bases de la anterior generación surgen:

1.Sistemas operativos de multiprogramación
2.Sistemas operativos de multiprocesamiento o multitarea
3.Sistemas operativos de tiempo compartido
4.Sistemas operativos de tiempo real

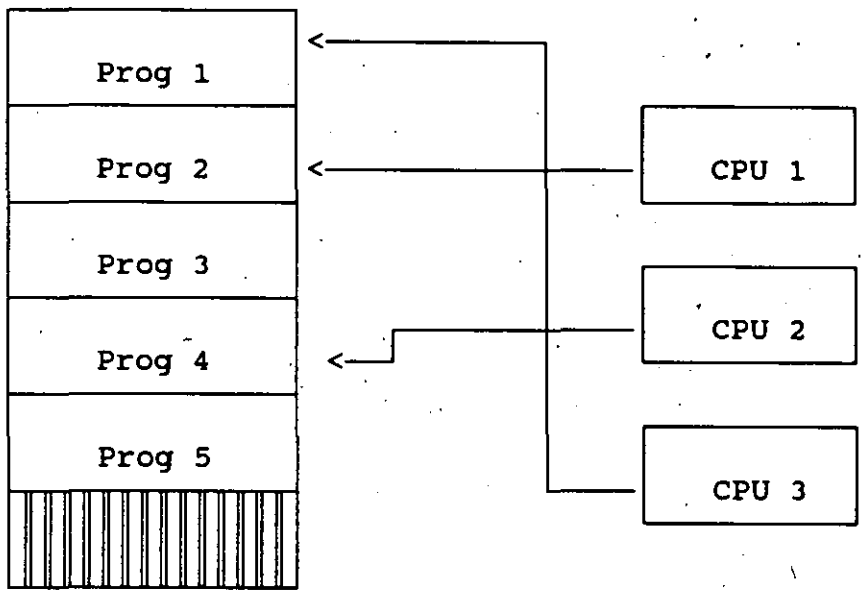
1.Sistemas operativos de multiprogramación

Son sistemas operativos que soportan varios procesos concurrentes permitiéndoles residir simultáneamente en la memoria principal. Los procesos compiten por el uso de los recursos del sistema de computación esto es para incrementar el uso de los mismos. Al número de procesos que compiten activamente por los recursos en un sistema multiprogramado se le llama grado de multiprogramación.



2.Sistemas operativos de multiprocesamiento o multitarea

Son sistemas operativos que pueden ejecutar varios procesos simultáneamente, ya que los sistemas computacionales de multiprocesamiento poseen varios procesadores con la finalidad de aumentar el poder de procesamiento del sistema.



3.Sistemas operativos de tiempo compartido

En estos sistemas operativos los usuarios podían acoplarse directamente con el computador a través de terminales parecidas a máquinas de escribir. Los sistemas de tiempo compartido operan en modo interactivo o conversacional con los usuarios. El usuario teclea una petición al computador, éste la procesa tan pronto como le resulta posible (generalmente en el transcurso de un segundo o menos) y la respuesta (si la hay) aparece tecleada enel terminal del usuario.

4.Sistemas operativos de tiempo real

En éstos sistemas los computadores fueron utilizados en el control de procesos industriales, como en la refinación de la gasolina.

Los sistemas de tiempo real se caracterizan por proveer una respuesta inmediata. En éstos sistemas es importante que se encuentren disponibles cuando sea necesario y que respondan rápidamente a que estén ocupados una gran parte del tiempo. Por ejemplo una lectura obtenida en una refinería de gasolina, indicando que las temperaturas están demasiado altas, puede demandar atención inmediata para poder evitar una explosión.

Tercera generación de los sistemas operativos (De la mitad de la década 1960 a la mitad de la década 1970)

Esta generación comenzó en forma efectiva en 1964, con la introducción de la familia de computadoras Sistema/360 de IBM.

Los sistemas operativos de esta generación eran sistemas de modos múltiples. Algunos de ellos soportan simultáneamente:

- Procesamiento por lotes
- Tiempo compartido
- Procesamiento en tiempo real
- Multiprocesamiento
- Multiprogramación

Estos sistemas introdujeron mayor complejidad a los ambientes computacionales; una complejidad a la cual, en un principio, no estaban acostumbrados los usuarios. Los sistemas operativos de la tercera generación representan un gran paso hacia adelante, pero también doloroso para muchos usuarios.

Cuarta generación de los sistemas operativos (De la mitad de la década 1970 a nuestros días)

Los computadores personales están equipados con interfaz para comunicación de datos y también sirven como terminales. El usuario de un sistema de la cuarta generación ya no se encuentra confinado a la comunicación con un solo computador en un modo de tiempo compartido. En lugar de esto, el usuario puede comunicarse con sistemas alejados geográficamente. Los problemas de seguridad se han incrementado mucho ahora que la información pasa a través de varios tipos vulnerables de líneas de comunicación.

Se tienen dos tipos de sistemas operativos:
a) Sistemas operativos de redes.
b) Sistemas operativos distribuidos.

a) Sistemas operativos de redes.

Son sistemas operativos que permiten que varios computadoras autónomas que se encuentran interconectados por una red de computadoras puedan establecer comunicaciones, de tal forma que se pueda procesar, copiar, enviar, recibir, etc., información desde cualquier punto de la red.

b) Sistemas operativos distribuidos.

Son sistemas operativos que ante el usuario aparentan estar en un solo lugar, es decir, se presentan como si fuera un sistema operativo individual en cada anfitrión; pero en realidad es un único sistema operativo homogéneo para la red.

Administración de los recursos

Un administrador es un conjunto de programas que administran los recursos del sistema de computo.
Funciones de un administrador.
*)Llevar registro del estado del recurso.
*)Decidir cuál proceso darle el recurso, es decir, hacer cumplir la política que determina quién obtiene el recurso, cuándo y por cuánto tiempo.
*)Asignarle el recurso.
*)Recuperar el recurso.

Funciones básicas de un sistema operativo.

*)Proporcionar una interfase sencilla de usar con el equipo.
*)Administrar los recursos, asignando y distribuyendo los recursos de equipo, software y datos a los diversos usuarios de una manera eficaz.
*)Optimizar la forma en la cual todos los recursos funcionan juntos para lograr algún objetivo deseado.

CARACTERISTICAS GENERALES DE LOS SISTEMAS OPERATIVOS

CARACTERISTICAS.

Concurrencia.

Es la existencia de varias actividades en un mismo período de tiempo.

Simultaneidad.

La existencia de varias actividades al mismo tiempo.

Eficiencia.

*) Tiempo de respuesta del sistema que sea mínimo.
*) Optimización en la utilización de recursos.
*) Tiempo promedio para la ejecución a procesos.

Tanto la optimización como el tiempo promedio van a depender de la máquina.

Compartibilidad.

Que varios procesos puedan compartir datos o bien, que sea necesario eliminar redundancias.

Confiabilidad.

Que las fallas que presente sean lo más esporádicas posibles.

Mantenimiento.

El sistema operativo debe estar hecho de una manera estructurada, modular y contar con una buena documentación para que sea fácil de mantener.

Pequeño.

Para que no ocupe demasiado espacio en memoria.

SISTEMA OPERATIVO COMO UNA MAQUINA EXTENDIDA.

La máquina extendida es la visión que se tiene del sistema operativo cuya función es la de presentarse al usuario como el equivalente a una máquina sencilla de utilizar que el hardware.

Por ejemplo para las operaciones que se tienen en un disco a nivel de máquina (máquina desnuda) se tiene lo siguiente:

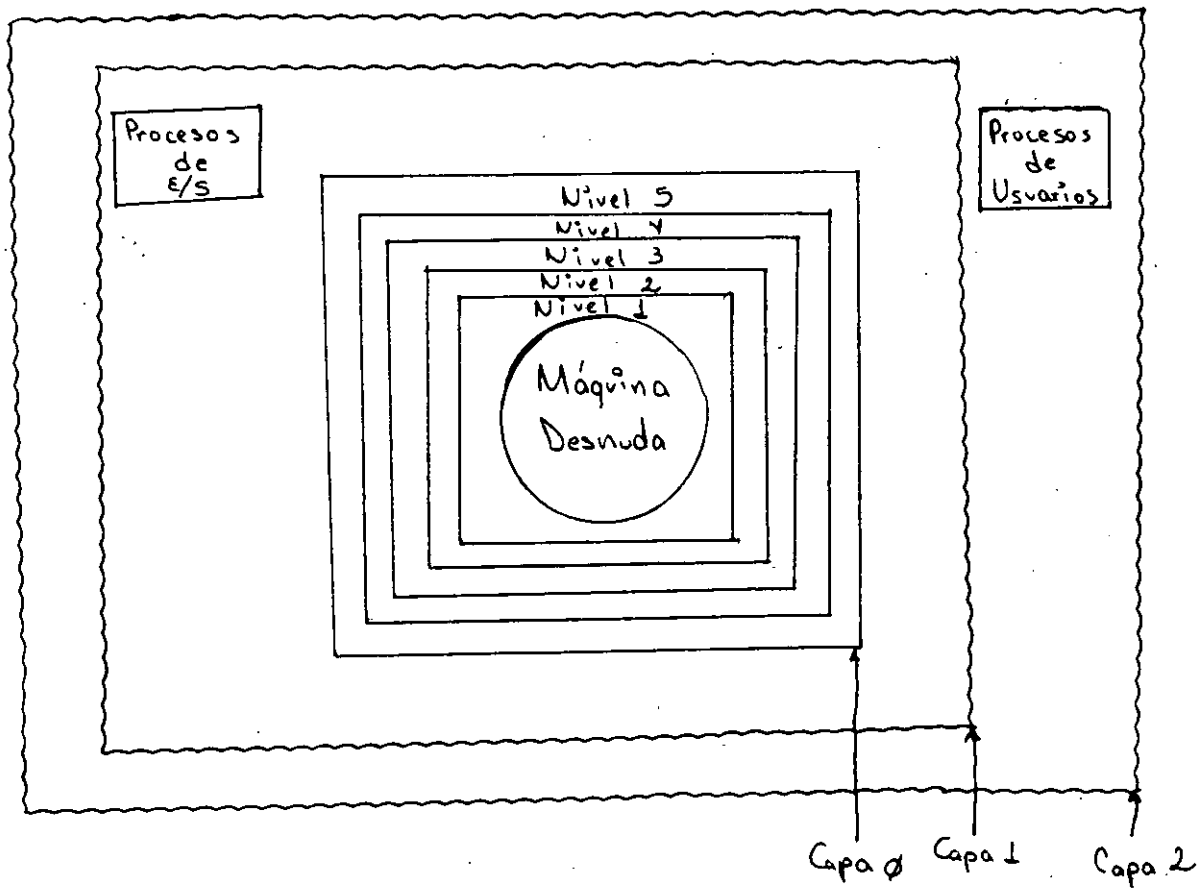
*)Dar la dirección del bloque del disco que se leerá/escribirá.
*)Dar el número de sectores por pista.
*)Dar el modo de grabación que se usa en el medio físico.
*)Dar el espacio entre sectores.
*)Verificar si el motor está apagado y si es así encenderlo.
*)Mover el brazo de la unidad de disco a la pista deseada.
*)Realizar la operación (Lectura/Escritura).
*)Apagar el motor.)
*)Revisar los campos de condición y error para comprobar que la operación se realizó bien.

A nivel de cualquier lenguaje de programación de alto nivel (se apoya en el sistema operativo)

*)Cada archivo puede abrirse para leerse o escribirse.
*)Leer/escribir el archivo.
*)Cerrar el archivo.

SISTEMA OPERATIVO COMO UNA MAQUINA JERARQUICA

Básicamente el concepto de máquina jerárquica en un sistema operativo, trata sobre la posición donde residen lógicamente los administradores de los recursos en relación de uno respecto a los otros.



Nivel 1: Parte inferior del administrador del procesador.
Nivel 2: Administrador de memoria.
Nivel 3: Parte superior del administrador del procesador.
Nivel 4: Administrador de dispositivos de E/S.
Nivel 5: Administrador de la información.

Administración de la memoria.

Funciones del administrador de memoria.

*)Llevar cuenta del estado de cada posición de la memoria principal.
*)Hacer cumplir la política de asignación de la memoria.
*)Asignar la memoria.
*)Recuperar la memoria.

Administración del procesador.

La administración del procesador se refiere al manejo del procesador (es) físico (s) específicamente para la asignación del (los) procesador (es) a los procesos.

Administración de la información.

La administración de la información se refiere al manejo que se hace con los archivos, en donde se involucra;
*)Operaciones involucradas con los archivos.
*)Capas conceptuales de un sistema de archivo.
*)Registros físicos y registros lógicos.
*)Organización y el acceso de archivos.

Administración de dispositivos de E/S.

En este tipo de administración se trata lo referente con:
*)Interfaces de E/S.
*)Controladores de E/S.
*)Dispositivos dedicados.
*)Dispositivos compartidos.
*)Dispositivos virtuales.

CARACTERISTICAS DE LA VAX/VMS 6000-210

ENTRADA AL SISTEMA VAX/VMS 6000-210

La VAX/VMS 6000-210 es una minicomputadora construida por Digital Equipment Corporation (DEC), con una longitud de 32 bits por palabra de memoria. El nombre VAX viene de 'Virtual Address eXtension', que significa extensión de direccionamiento virtual y esto viene porque la VAX tiene mayor capacidad de direccionamiento virtual que sus antecesoras las PDP. La capacidad de memoria virtual le permite procesar programas más grandes que el tamaño de su memoria física, gracias al concepto de paginación.

Los procesadores VAX manejan palabras de 32 bits (LONGWORD), lo cual les permite direccionar hasta 4.3 Gigabytes de memoria virtual. Las páginas de memoria física, virtual y los bloques en disco que maneja VMS son de 512 bytes. Por lo tanto al referirnos al almacenamiento en disco en VMS nos expresaremos en términos de bloques y no en Kbytes.

En esta computadora se encuentra instalado el sistema operativo VMS 5.4-2 con interfase sencilla y amigable para el usuario, para la programación, graficación e intercambio de datos lo cual lo hace muy portable.

Características del Hardware

- Tecnología CMOS
- Memoria RAM de 32 MB
  (Expandible hasta 256 Mbytes)
- Dos discos duros RA90 de 1.2 Gbytes c/u.
- Dos discos duros RA92 de 1.5 Gbytes c/u.
- Dos discos duros RA70 de .5 Gbytes c/u.
- Una unidad de cartucho TK70
- Una unidad de cinta TU81
- Un plotter LG02 de 600 líneas/min
- Una impresora rápida LP37 de 1200 líneas/min.
- Dos DECWRITER LA-120

Software Instalado

- Compiladores
  BASIC, FORTRAN, COBOL, C, PASCAL.
- Utilerías
  FMS (Sistema Manejador de Formas)
  RMS (Servicio de Manejo de Registros)
  MAIL, PHONE.
- Manejadores de base de datos
  ORACLE
  RDB,ADABAS
- Procesador de Texto
  RUNOFF
- Editores
  TPU, EDT
- Comandos desarrollados por CECAFI
  MAKE (permite actualizar dependientes en caso de que se modifique alguna fuente)
  KILLTREE (este comando permite borrar árboles de subdirectorios completos)
  RANGE (permite determinar cuántos registros lógicos tiene un archivo determinado)
  CPQ (permite correr programas y paquetes en forma muy amigable para el usuario)

Comandos de terminal

La terminal es el medio a través del cual nos comunicamos con la computadora, es por ello que conviene conocer algunas de sus características. A continuación presentamos algunos de los comenados estándar para cualquier terminal sin meternos en lo específico de una VT100 o VT420 que son con las que cuenta el centro.

CTRL/C y CTRL/Y
. Si se dan durante un comando de entrada, cancela el procesamiento del comando.
. Antes de una sesión de terminal, inicia la secuencia del Login.
. Interrumpe el comando o la ejecución de un programa y regresa el control al intérprete de comandos.

CTRL/I
. Avanza el cursor a la siguiente marca del tabulador.

CTRL/K
. Avanza la línea actual a la siguiente marca del tabulador.

CTRL/L
. Avanza al final de la página.

CTRL/O
. Alternativamente, suprime y continúa mostrando un despliegue de información en la terminal.

CTRL/Q
. Restaura el despliegue de información en la terminal que fue suspendida por CTRL/S.

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

CTRL/U
. Suprime la línea actual.

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

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

CTRL/W
. Refresca la pantalla con la última página del archivo con que se está trabajando.

ESC
. Esta tecla permite proporcionar una secuencia de ESCape para modificar las características de la pantalla o auxiliarnos en el manejo de la misma.

Mensajes del sistema

Lo primero que se verá al entrar al sistema será un mensaje de bienvenida definido por el administrador del sistema, posteriormente se podrán desplegar mensajes del sistema al usuario, tales como advertencias, mensajes varios , etc...

Aparecerá el PROMPT (indicador) de DCL (Digital Command Lenguage), el cual es un **$**. Cuando esto sucede se dice que el usuario ha entrado en sesión lo que significa que se puede hacer uso de los recursos de la computadora.

Dificultades de acceso

Los mensajes que podrá dar la máquina en caso de una dificultad de acceso son los siguientes:

a) Que aparezca un mensaje que diga:

USER AUTHORIZATION FAILURE, esto es que se cometió un error al poner el USERNAME o el PASSWORD por lo que tendrá que iniciar nuevamente el proceso de entrada.

Esta clase de errores solo puede ser repetido tres veces, ya que sin no se desconectará del nodo de ATZIN y aparecerá el prompt del servidor, lo cual indicara que esta desconectado.

b) Si la computadora no responde puede ser que en ese momento el sistema esté fuera de servicio momentáneamente ('caído') por lo que deberá esperar el mensaje correspondiente al reinicio de operaciones.

c) Si se entra en sesión pero no permite ver el directorio, ni los archivos puede ser que el disco al que pertenece la clave no esté disponible, por lo que deberá terminar la sesión y preguntar a que hora estará disponible el disco.

PRIMERAS INSTRUCCIONES DE COMANDO

INSTRUCCIONES BASICAS

El sistema vax/vms 6000-210 utiliza un símbolo para especificar sus distintos modos de operación, entre estos se tiene:

COMANDOS DEL SISTEMA (Nivel DCL) Caracterizada por un signo de pesos a la izquierda.

$

COMANDOS DEL EDITOR (Modo línea) Caracterizada por un asterisco a la izquierda.

_*

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

[EOB]

Instrucciones básicas

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

DISPOSITIVO:[RUTA]NOMBRE-DE-ARCHIVO.EXTENSION;VERSIÓN

El dispositivo es el nombre donde está físicamente guardado el archivo esto es disco, cinta, tk, etc. Las primeras dos letras son en código del dispositivo,: DU (Magnetic Unit) para disco, TX para impresora o terminal conectada a un puerto serial de la VAX, MU (Magnetic Unit) para cinta magnética. Después está el nombre del controlador que consta de una letra (A,B,C, etc.) y el número que ocupa dentro del controlador (0,1,2,3, etc.) por lo que para la configuración actual de la VAX los nombres son:

DUB0:	Disco cero controlador B
DUB1:	Disco uno controlador B
DUB2:	Disco dos controlador B
DUB3:	Disco tres controlador B

DUA10:	Disco uno controlador A
DUA11:	Disco dos controlador A
MUA0:	Unidad de cinta convencional.
MUB6:	Unidad de cinta compacta TK70.
TXA2:	Impresora DECWRITER
TXA4:	Impresora DECWRITER
TXA0:	Impresora graficadora (Plotter)
TXA1:	Para la impresora Laser
LIA0:	Puerto paralelo. Impresora de línea.
LTAxx:	Terminales conectadas a los servidores

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

```
BASICEJEMPLO.BAS
FORTRANEJEMPLO.FOR
COBOLEJEMPLO.COB
PASCALEJEMPLO.PAS
```

Ejemplos de nombres de archivos:

```
DBA1:[CLASE]EJEMPLO.BAS;3
MUA0:[CINTA]TRABAJO.FOR;6
DBA2:[JUANITO]DATOS.PAS;4
```

Consideraciones

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

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

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

$ DIR <return>

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

$ CREATE/DIRECTORY especificación del directorio

donde especificación es [directorio.subdirectorio]

Ejemplo:

$ CREATE/DIR [CURSOS.EGK] <return>

Se crea el subdirectorio [.EGK] (iniciales de Ernesto Gutiérrez Kuri) en el directorio [CURSOS]

$ CREATE/DIR [AMH200.BASIC]

Se crea el subdirectorio BASIC en la clave(directorio) [AMH200].

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

$ SET DEFAULT [directorio.subdirectorio]

$ SET DEF [CURSOS.EGK] <return>

$ SET DEF [.BASIC] <return>

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

$ SET DEF [CURSOS] si nuestra clave es CURSOS o
$ SET DEF [AMH200] si nuestra clave es AMH200, etc.

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

```
$ SHOW DEFAULT
```

que proporcionará la ruta completa en donde se encuentra el usuario.

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

```
$ SHOW TIME que proporciona la fecha y la hora.
$ SHOW QUOTA que da el UIC, cuantos bloques se han utilizado
  cuántos quedan disponibles y en que disco se está trabajando.

$ SHOW DAYTIME es equivalente a $ SHOW TIME.
```

LA AYUDA DE DCL

INSTRUCCION HELP

Existe una gama muy amplia de comandos de sistema que se pueden usar en el sistema VAX 6210 tales como SHOW, SET, etc... Todos estos están resumidos en una biblioteca a la que se tiene acceso mediante el comando HELP, en el cual están los formatos de los comandos de DCL. Algunas instrucciones están restringidas a la mayoría de los usuarios pero aun así hay gran versatilidad. El formato de HELP es:

```
$ HELP  <return>
```

A continuación aparece un desplegado con todos los comandos que existen en DCL y se puede pedir su sintaxis escribiendo el comando después del mensaje que dice Topic?. La computadora dará al usuario una pequeña explicación del comando y proporcionará información adicional que puede explicar. Si así se desea, el usuario deberá escribirla después de la palabra Subtopic? La forma de salirse del HELP es mediante <return> sucesivos hasta que aparezca el signo de $. Es conveniente que las primeras veces que se use la VAX se compenetre en el uso de este comando de DCL.

```
$ HELP  <return>

  Information available:
  ACCOUNTING ALLOCATE   ANALYZE     APPEND
  ASSIGN     ATTACH     BACKUP      BASIC
  BLISS      CANCEL     CC          CLOSE
  COBOL      CONTINUE   CONVERT     COPY
  CORAL      CREATE     DBO         DDL
  DEALLOCATE DEASSIGN   DEBUG       DECK
  DEFINE     DELETE     DEPOSIT     DIFFERENCES
  DIRECTORY  DISMOUNT   DUMP        EDIT
  EOD        EOJ        Errors
  EXAMINE    EXIT       FDL         FORTRAN
  GOTO       HELP       IF          INITIALIZE
  INQUIRE    JOB        Lexical
  LIBRARY    LINK       Login       LOGOUT
  MACRO      MAIL       MCR         MERGE
  MESSAGE    MONITOR    MOUNT       ON
  OPEN       PASCAL     PASSWORD    PATCH
  PHONE      PLI        PRINT       Procedure
  PURGE      Queues     READ        RENAME
  REPLY      REQUEST    RMS         RTL
  RUN        RUNOFF     SEARCH      SET
  SHOW       SORT       SPAWN
  Specify    START      STOP        SUBMIT
  Symbol__Assign         SYNCHRONIZE
  System     TECO       TYPE        UNLOCK
  WAIT       WRITE

Topic?  dir  <return>

DIRECTORY

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

   Format:

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

  Additional information available:

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

```
DIRECTORY Subtopic? /full  <return>

DIRECTORY

  /FULL

   Lists full file attributes with each file.
   The /FULL qualifier overrides the default brief listing format.

DIRECTORY Subtopic?  <return>
Topic?  <return>
$
```

INTRODUCCION AL EDITOR EDT

Comando EDT

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

Modo línea

```
$ Se escribe EDIT/EDT <NOMBRE DEL ARCHIVO>

$ EDIT/EDT <return><return> IMPLICA PRESIONAR LA TECLA RETURN !!!
Y aparece en el display...
$-file:
```

Ahora se tiene la capacidad de dar el nombre y la extensión al programa, lo cual se deberá hacer y presionar <return>.
En seguida nos aparece el letrero:

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

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

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

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

```
$ EDT AA. <return>

Input file does not exist
[EOB]
_* HELP  <return>
HELP
You can get help on a topic by typing
     HELP topic subtopic subsubtopic...
```

```
A topic can have one of the following forms:
     1. An alphanumeric string
           (e.g. a command name, option, etc.)
     2. The match-all or wild card symbol (_*)
     Examples: HELP SUBSTITUTE NEXT
               HELP CHANGE SUBCOMMAND
               HELP CH
If a topic is abbreviated, the text for all topics
which match the abbreviation is displayed.
Additional information available:
CHANGE      CLEAR       COPY        DEFINE      DELETE
EXIT        FILL        FIND        HELP        INCLUDE
INSERT      JOURNAL     KEYPAD      MOVE        PRINT
QUIT        RANGE       REPLACE     RESEQUENCE  SET
SHOW        SUBSTITUTE  TABS        TYPE        WRITE
_* EXIT

$ DISK$USUARIOS1:[CHARLY]AA.;1 No lines
```

Los comandos más utilizados son:

_* INSERT

Su formato general es:

_* INSERT [rango] [; línea a ser insertada]

Donde rango puede ser:

1) Un número de línea

2) La palabra BEGIN que indica el inicio del archivo

3) La palabra END que indica el fin del archivo

4) Cualquier combinación de éstas separadas por dos puntos (:) o la palabra THRU.

5) La palabra WHOLE que significa todo

6) Rango + número

7) Rango - número

8) 'String' a buscar

9) LAST

10) (línea actual)

11) Número de línea cuantas más

12) BEFORE

13) REST

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

_* I 8

agrega líneas antes de la línea numerada con 8

_* I 8; B = 2 _* A

inserta la línea que aparece en el comando antes de la 8.

Para salirse de INSERT se oprimen simultaneamente CRTL y Z

_* INSERT

Permite introducir el archivo (programa), el cursor Inicia en la columna 10 aproximadamente pero es la primera columna del archivo. Cada vez que damos <return> el cursor pasa a la siguiente línea. Para terminar hay que utilizar CTRL/Z.

_* SUBSTITUTE

Permite corregir el archivo.

Su forma general es:

_* SUBSTITUTE/string1/string2/[rango][BRIEF:n][/QUERY]

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

_* S/error/error/5

En este caso solo la línea 5 se modifica

_* S/alumno/estudiante/WHOLE/QUERY

Se desea que efectúe las substituciones en todo el archivo pero que vaya preguntado antes de hacerlas.

_* REPLACE

Permite substituir líneas existentes por las que se van a teclear a continuación; al terminar hay que dar CTRL/Z.

_* REPLACE [rango][;línea a ser insertada]
_* MOVE

Mueve líneas de texto de un lugar a otro.

_* MOVE [rango1] TO [rango2] [/QUERY]

donde rango1 es el conjunto de líneas que se quieren mover y rango2 es el lugar donde se quieren que estén.

_* COPY

Esta instrucción permite duplicar un conjunto de líneas en otro lugar del archivo.

_* COPY [rango1] TO [rango2] [/QUERY] [/DUPLICATE:n]

Donde n es el número de veces que se desea duplicar el texto.

_* RESEQUENCE
Cuando se desea que la numeración interna del archivo se normalice o modifique, se puede usar ésta instrucción.

_* RESEQUENCE [rango] [/SEQUENCE] [:inicio [:incremento]]
_* RES 2:5/SEQ :2:3

Se está pidiendo que el bloque de líneas que va del 2 al 5 se renumere iniciando en 2 y con incrementos de 3 en 3. Se hace notar que entre 2 y 5 pueden existir una gran cantidad de líneas.

_* TYPE

Permite ver en la terminal el archivo, con la numeración interna proporcionada por el Editor.

_* TYPE [rango] [/BRIEF:n] [/STAY]

donde el calificador /STAY ocasiona que el apuntador de línea no avance de su posición actual.

_* TYPE 'string'

Busca la ocurrencia del string y nos despliega la línea; si se añade ALL desplegará todas las líneas que contengan el string

_* FIND

Este comando permite ubicarse en una línea, esto es, el apuntador de líneas se mueve a ésta.

_* FIND [rango]
_* DELETE

Permite borrar líneas del archivo

_* DELETE [rango]
_* EXIT

Permite salirse del Editor salvando el archivo en el disco.

_* QUIT

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

_* INCLUDE

Con este comando se puede traer un archivo en el disco e incluirlo en el archivo que se está editando.

_* INCLUDE nombre-de-archivo [rango]

_* PRINT

Manda al espacio en disco una copia del archivo que se está editando, pero incluye la numeración interna.

_* PRINT nombre-de-archivo [rango]
_* WRITE

Crea un archivo en nuestro espacio en disco, sirve para crear protecciones temporales cuando el sistema puede tener fallas continuas y nuestros archivos son muy grandes.

_* WRITE nombre-de-archivo [rango] [/SEQ [:inicio[:inc.]]]
_* CHANGE

Este comando permite cambiarse al otro modo del Editor pero que consume más recursos de computadora. Estaremos en modo INSERT.

_*C <return>

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

_*QUIT <return> o
_*EXIT <return>

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

Modo caracter

Las FUNCIONES EDITOR en modo caracter se describirán en el siguiente cuadro y se refieren al KEYPAD(teclado auxiliar situado a la derecha del teclado).

FUNCION	CARACTERISTICAS
PF 1	Permite efectuar la operación alterna de cada tecla.
PF 2	Despliega todas las funciones del KEYPAD (HELP)
PF 3	Se posiciona en el siguiente STRING que se haya pedido que busque o da el mensaje cuando no se encontró.
PF 1 PF 3	Aparece un mensaje que dice SEARCH FOR:, se puede dar cualquier STRING que se deseé buscar y necesita darle la dirección de búsqueda.
PF 4	Borra línea por línea, estando situado al principio de ésta
PF 1 PF 4	Imprime la última línea que se haya borrado con PF 4 estando situado al lado izquierdo del display
,	Borra caracter por caracter de izquierda a derecha
PF1 ,	Imprime en el display el último caracter que se haya borrado con ,
__	Borra palabra por palabra de izquierda a derecha, esto es de blanco a blanco, los caracteres entre espacios
PF1 __	Imprime en el display la última palabra que se haya borrado con __
7	Avanza a la siguiente página, y depende de la dirección actual encendida

PF 1 7	Permite ejecutar un comando del editor en modo línea, hay que oprimir <enter>
8	Avanza secciones de 16 renglones
PF 1 8	El texto seleccionado es formateado para llenar el número de columnas que se fije con SET WRAP (modo línea).
9	El párrafo seleccionado es guardado al final del BUFFER
PF 1 9	Substituye el texto seleccionado por el texto contenido en el BUFFER.
4	Indica que todos los movimiento deseados son hacia el final del archivo
PF 1 4	Se sitúa en la parte inferior del archivo
5	Indica que todos los movimientos deseados son hacia el inicio del archivo
PF 1 5	Se sitúa en la parte superior del archivo
6	Corta el párrafo que se haya seleccionado y lo guarda en un BUFFER.
PF1 6	Imprime el último párrafo que se haya seleccionado y guardado en el BUFFER
1	avanza una palabra (en la dirección seleccionada
PF 1 1	Abre una línea donde esté el cursor para insertar texto.
2	Se posiciona al final de la línea donde se encuentre el cursor
PF 1 2	Borra todos los caracteres desde donde está el cursor hasta el fin de la línea
3	Avanza un caracter (en la dirección seleccionada)
PF1 3	Permite insertar un caracter ASCII

.	Inicia la selección de un párrafo que se quiera borrar o mover de lugar, abarcándolo con las flechas guía.
PF1 .	Cancela la selección del párrafo
ENTER	Permite que el comando de Editor modo línea indicado después de PF 1 7 sea ejecutado por la computadora
PF1 ENTER	Substituye el STRING buscado por el texto en el BUFFER y busca la siguiente ocurrencia del STRING

El siguiente diagrama muestra la estructura del KEYPAD

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

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

IMPRESION DE ARCHIVOS

INSTRUCCIONES PRINT Y COPY

Instrucción PRINT

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

```
$ PRINT nombre__del__programa__o__archivo.extensión;versión
```

Después el sistema genera un aviso de entrada de impresión:

JOB FILENAME (queue SYS$PRINT,entry nnn) started on LIA0.

Donde nnn es el numero de listado de impresión que le asignó al programa. Cuando el listado ha terminado de salir por la impresora, la computadora enviará el mensaje correspondiente. Si se desea escribir varios archivos bajo la misma carátula de salida lo que se tiene que hacer es asignarlos después de la instrucción PRINT separados por comas(,) o por mas(+). El formato es:

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

Si no se escribe la versión, la computadora entiende que se quiere mandar a imprimir la última versión. Está restringido por sistema y no tiene sentido mandar a imprimir archivos del tipo .OBJ y .EXE.

Instruccción COPY

Mediante la instrucción COPY se puede asignar a un solo archivo un conjunto de ellos, o copiar archivos de un subdirectorio a otro. El formato es:

```
$COPY                                          <return>
$__from: que__archivo__vas__a__asignar         <return>
$__to: a__que__archivo__lo__vas__a__asignar    <return>
```

En algunos casos el sistema enviará un mensaje de incompatibilidad de archivos pero no tiene trascendencia. En la parte de $_to: del formato se debe asignar un nombre con la extensión que se desee, este nuevo archivo estará en el directorio personal mediante el nombre que se designó y contendrá todos los archivos que se determinaron en el $_from:.

INSTRUCCIONES PURGE Y DELETE

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

INSTRUCCION PURGE

Esta instrucción depura el directorio dejando únicamente la última versión de cada programa que se tenga en el directorio. Hay que tener mucho cuidado al utilizar esta instrucción ya que muchas veces la versión que sirve no es la última y si se ejecuta la instrucción PURGE el sistema eliminará la versión útil. El formato es:
$ PURGE <return>
Aparecerá el signo de $.

INSTRUCCION DELETE

Esta instrucción permite borrar un determinado archivo o familia de ellos. El formato es:

$ DELETE nombre__del__archivo.extensión;versión <return>

Puede aparecer el signo de $ o un mensaje que indique el hecho de tratar de borrar un archivo que no existe. O que se tiene privilegios para borrar el archivo (como son los subdirectorios) El asterisco (_*) tiene la particularidad de generalizar como sigue:

$ DEL _*.NNN;_*

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

$ DEL _*._*;N

Borra el archivos cuya versión sea la indicada

$ DEL NNNN._*;_*

Borra todas las versiones y extensiones del archivo indicado. Las instrucciones para borrar pueden ser las mas fáciles de ejecutar, pero también son las mas peligrosas ya que se puede llegar a borrar archivos o programas que posteriormente harán falta.
El uso del (_*) como generalizador (wild card) es válido en cualquier instrucción de DCL relacionada con archivos. Otro caracter es (_%) que generaliza un solo caracter. Una instrucción que nos permite cambiar el nombre a archivos ya existentes es el $ **RENAME** nombre-viejo nombre-viejo.

INSTRUCCION LOGOUT

Esta instrucción sirve para despedirse del sistema VAX 6210, y se debe utilizar al finalizar la sesión de terminal. Se puede utilizar el Comando LOGOUT, su abreviatura LOG o el símbolo LO. Formatos:

```
$ LOGOUT <return>
$ LOG  <return>
$ LO <return>
```

Aparecerá un mensaje de despedida del sistema y la conexión con VAX acaba de concluir.

SIMBOLOS Y PROCEDIMIENTOS DE COMANDOS

SIMBOLOS LOCALES Y GLOBALES

Los símbolos son nombres de variables, de 1 a 9 caracteres, en las cuales podemos guardar números reales o enteros o strings (que pueden ser comandos de DCL). Los símbolos pueden ser locales a un procedimiento o pueden ser globales a todos los procedimientos y comandos ejecutados en la sesión. Para almacenar datos en los símbolos se deben seguir las siguientes reglas:

```
Para valores numéricos locales:
            SIMBOLO = valor
Para valores numéricos globales:
            SIMBOLO == valor
Para strings locales:
            SIMBOLO := string
Para string globales:
            SIMBOLO :== string
```

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

```
$AUXILIO:==HELP
```

Cada vez que se escriba AUXILIO, la computadora ejecutará el comando HELP. Si dentro del nombre del símbolo se inserta un asterisco, entonces todos los caracteres que le siguen son opcionales. Ejemplo:

```
$AUX_*ILIO:==HELP
```

Con solo teclear AUX se ejecutará el comando HELP. A nivel DCL se permite ejecutar operaciones con los símbolos. Estas operaciones son: suma, resta, producto y división para los valores numéricos; búsqueda, extracción y concatenación, y substitución de contenido en los strings.

PROCEDIMIENTOS DE COMANDOS

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

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

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

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

Se puede ejecutarlo también mediante un SUBMIT, que ocasiona que el proceso se ejecute en modo BATCH (no interactivo). También se puede ejecutarlo desde otro procedimiento de comandos.

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

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

Control de Entrada/Salida de los procedimientos de Comandos

Cuando se inicia una sesión, el sistema operativo ejecuta un proceso en el cual se establecen las equivalencias iniciales de los siguientes nombres lógicos:

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

Cuando se inicia una sesión, SYS$INPUT se encuentra asignado a la terminal en la cual se está trabajando. Cuando se ejecuta un procedimiento de comandos, se establece una nueva equivalencia para el nombre lógico SYS$INPUT, la cual será el propio archivo que contiene el procedimiento de comandos. Si se anidan procedimientos de comandos, es decir si es ejecutado un procedimiento dentro de otro procedimiento de comandos, SYS$INPUT es asignado al archivo que corresponde al procedimiento que se está ejecutando en ese momento. SYS$COMMAND es hecho equivalente al nivel de comandos en que se encuentre: si se ejecuta un procedimiento de comandos interactivamente, este nombre lógico es asignado a la terminal, mientras que si ejecutado desde batch es asignado a la cola de procesos tipo BATCH.

Definición de teclas

En cada una de nuestras terminales tenemos, en la parte derecha del mismo un teclado numérico llamado KEYPAD el cual puede tener un uso numérico o de teclas especiales, para ello es necesario configurar nuestra terminal para que trabaje de modo comando, para que una vez hecho esto sea posible utilizar las teclas del keypad como un arma mas en el uso del sistema operativo, ahorrándonos tiempo y trabajo. Con el comando :

```
$set terminal/application
```

es posible usar el keypad como no numérico es decir para uso de teclas especiales las cuales se definen con el siguiente comando:

```
$define /key  <nombre de la tecla> <comando>
```

donde : nombre de tecla pueden ser los siguientes nombres predefinidos:

KP0	COMMA	PF1
KP1	MINUS	PF2
KP2	ENTER	PF3
.	PERIOD	PF4
.		
.		
KP9		

Por ejemplo para definir la tecla PF1 para que una vez que se apriete aparezca el comando DIR tenemos:

```
$define /key PF1 "dir"
```

con esto aparecerá el comando dir en la pantalla teniendo que apretar la tecla <enter> para que se ejecute. Por otra parte, si queremos que al apretar la tecla se ejecute automáticamente el comando es necesario poner otra opción del comando /terminate

```
$def /key PF1 "dir"/terminate
```

Con ello se ejecutara inmediatamente después de apretada la tecla el comando "DIR". Así como definimos la tecla PF1 se pueden definir las otras teclas lo cual puede ser de mucha ayuda por ejemplo

```
$home:==set def sys$input
$define/key KP0 "home"/teminate
```

En este caso se al presionar la tecla 0 del keypad nos vamos a cambiar al directorio en el cual estamos al iniciar la sesión, es decir a nuestro directorio raíz.

SISTEMA OPERATIVO MS-DOS.

C E C A F I

FACULTAD DE INGENIERIA

ABRIL 1991

JORGE ALBERTO M. ROJAS ARIAS

SISTEMA OPERATIVO MS-DOS.

PREFACIO

El presente material tiene la finalidad de instruir al lector con respecto al manejo de las computadoras personales (PC), es decir explica la forma de comenzar a trabajar con dichas máquinas, proporcionando una explicación de las instrucciones ó comandos con los cuales se manipulan las mismas.

De antemano se advierte que este material fué elaborado por los recursos humanos del Centro de Cálculo de la Facultad de Ingeniería (CECAFI) y a pesar de que ha tenido múltiples revisiones, no se encuentra excento de errores, así como de profundidad y amplia explicación de algún tópico en especial; es por ello que se pide al lector que disculpe todos los errores que encuentre en el presente material, agradeciendo que todos los comentarios, sugerencias, observaciones y críticas que tenga sobre el mismo, se hagan saber al Centro de Cálculo a través de sus instructores y/o personas que lo representan.

Así mismo, quiero dar las gracias a el Ing. Laura Sandoval Montaño por la revisión técnica y la colaboración en general para la realización de este material.

JORGE ALBERTO M. ROJAS ARIAS.

INDICE GENERAL

I. INTRODUCCION.

Antes de comenzar propiamente con el manejo de una microcomputadora a través del Sistema Operativo MS-DOS,es necesarios tener presente algunos conceptos básicos que son indispensables para el mejor entendimiento de lo que se hace y de lo que se puede hacer con una computadora.

SISTEMA DE COMPUTO.

Entenderemos como un sistema de cómputo como aquella herramienta que nos permitirá hacer de una forma más rápida y eficiente aquellas tareas que el hombre realiza manualmente.

Los elementos básicos que componen el sistema de cómputo son el "hardware" y el "software". El Primero lo constituyen los componentes mecánicos y/o electrónicos que forman el equipo físico, y el software lo forman el sistema operativo y los programas o paquetes de aplicación.

Algunos elementos del "hardware" son : el microprocesador, el almacenamiento en disco flexible y duro, los dispositivos de entrada y salida y la impresora.

Con respecto al "software" tenemos los siguientes : sistemas operativos, programas de aplicación, manejadores de bases de datos (DBASE,INFORMIX), hojas de cálculo (LOTUS,WORD), étc. entre otros.

COMPONENTES DEL COMPUTADOR.

Los elementos que conforman un sistema computador son :

1) Gabinete Principal.

2) Monitor de Video.

3) Teclado.

MICROPROCESADOR.

Un microprocesador es aquella parte de la computadora que se encarga de realizar todas las operaciones y en general de procesar toda la información, como son cálculos, generación de gráficas, dibujos, étc.

TIPOS DE MICROCOMPUTADORAS.

Las computadoras personales (PC's) se clasifican en dos grandes grupos, dependiendo del microporcesador que tengan y son:

XT
AT

Microcomputadoras XT : son aquellas microcomputadoras que tienen un microprocesador cuyo número que los representan son : 8088, 8086 y 8087. Las siglas XT vienen de las palabras eXtended Tecnology.

Microcomputadoras AT : son aquellas microcomputadoras cuyos microporcesadores pueden ser : 80286, 80386 y 80486. Las siglas AT provienen de Advanced Tecnology.

Ahora bien, la rapidez de una computadora personal se debe a al microprocesador que traen y es por eso que es muy común que escuchemos hablar de equipos XT's ó equipos AT's, ya que en general todas las Microcomputadoras AT's son mucho más rápidas que las XT's.

PROGRAMA

Programas, frecuentemente llamados programas de aplicación, aplicaciones o software, son una serie de instrucciones escritas en lenguajes de computadora. Estas instrucciones se guardan en archivos y le indican a su computadora que realice una labor.

ARCHIVO

Un archivo es un conjunto de información relacionada, tal como el contenido de una carpeta en el cajón de un escritorio. Las carpetas de archivo, por ejemplo, pueden contener cartas de negocios, memorandums de oficina o datos de las ventas mensuales. Los archivos de su disco también pueden contener cartas, memorandums o datos. Por ejemplo, su disco que usted tiene pueden contener archivos que usted ha creado, o archivos que fueron suministrados con el disco.

NOMBRE DE ARCHIVO.

De la misma forma que cada carpeta en un archivador tiene una etiqueta, cada archivo en un disco tiene un nombre. Este nombre tiene dos partes : un nombre de archivo y una extensión. Un nombre de archivo puede tener desde uno hasta ocho caracteres de longitud, y puede ser escrito en letras mayúsculas o minúsculas. MS-DOS convierte los nombres de archivos automáticamente a letras mayúsculas.

Las extensiones de los nombres de archivo constan de un punto seguido de uno, dos o tres caracteres. Las extensiones son opcionales, pero es conveniente utilizarlas ya que son útiles para describirle el contenido de un archivo a usted y a MS-DOS. Ejemplo:

INSCRIPCIONES.DAT

Donde : INSCRIPCIONES es el nombre del archivo y
DAT es la extensión del archivo.

DIRECTORIO.

Un directorio es un índice del contenido de un disco. Este contiene los nombres de los archivos, sus tamaños y las fechas en que fueron modificados por última vez.

ETIQUETA DE VOLUMEN.

Cuando usted utiliza un disco nuevo, puede ponerle una etiqueta en la parte exterior para ayudarle a identificar su contenido. También puede darle a cada uno de sus discos un nombre interno, llamado **etiqueta de volumen.**

UNIDAD DE DISCO.

Para utilizar los archivos que se encuentran almacenados en un disco flexible, primero hay que insertar el disco en una unidad de disco flexible. A las unidades de disco flexible muchas veces se les llama la unidad A y la unidad B; al disco fijo ó duro, normalmente instalado dentro de su computadora, se la llama la unidad C.

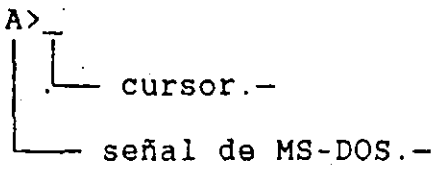
COMANDO.

De la misma forma que usted ejecutará programas para crear y actualizar archivos que contiene sus datos, también necesitará

algunos programas especiales, llamados comandos de MS-DOS, que le permiten trabajar con archivos enteros. Cuando usted escribe ciertos comandos de MS-DOS, le está pidiendo a la computadora que realice tareas.

LA UNIDAD PREDETERMINADA.

Si usted no especifica el nombre de la unidad cuando escribe el nombre del archivo, MS-DOS automáticamente busca el archivo en la unidad de disco predeterminada. La unidad predeterminada es el lugar donde MS-DOS busca primero cuando usted escribe un comando. Para hacerle saber que está listo para recibir un comando, MS-DOS muestra un símbolo llamado señal que contiene la letra de la unidad de disco predeterminada seguida del signo mayor que (>). A la derecha de la señal está el cursor, que es un pequeño bloque o barra parpadeante que muestra donde aparecerá el siguiente carácter que usted escribe. He aquí un ejemplo de una señal de MS-DOS y el cursor :



TIPOS DE DISKETTES

El diagrama siguiente indica los tipos de diskettes utilizados para leer y grabar información.

Tamaño (pulgadas)	Descripción	Capacidad (Bytes)
5.25	Una cara, doble densidad	160KB/180KB
5.25	Dos caras, doble densidad	320KB/360KB
5.25	Alta capacidad, doble densidad	1.2 MB
3.5	Dos caras	720 KB
3.5	Dos caras	1.44 MB

SISTEMA OPERATIVO.

Definiremos a un sistema operativo como al conjunto de programas que traducen sus comandos para la computadora, ayudándole a realizar tareas tales como crear archivos, ejecutar programas e imprimir documentos. En otras palabras, un sistema operativo es el administrador del sistema de cómputo que nos va a permitir hacer uso de los recursos de la computadora.

DISPOSITIVOS.

Cuando usted utiliza una computadora, introduce información (información de entrada) y obtiene un resultado (información de salida). La computadora utiliza equipos llamados dispositivos para recibir y enviar información.

MEMORIA.

La memoria es el lugar en su computadora donde la información se utiliza activamente. Al ejecutar un programa, MS-DOS almacena ese programa y los archivos que utiliza en la memoria disponible de la computadora. Algunos programas y archivos utilizan más memoria que otros, según el tamaño y la complejidad.

II. INICIO DE SESION EN UNA PC.

COMO INICIAR MS-DOS.

Para iniciar MS-DOS, siga estos pasos (estos pasos trabajan tanto en computadoras que tengan disco fijo (duro) como disco flexible):

1. Primero, asegúrese que su computadora esté apagada.
2. Retire el disco original de MS-DOS de la cubierta protectora.
3. Inserte el disco en la unidad A.
4. Cierre la puerta de la unidad de disco.
5. Encienda su monitor y su computadora.

La luz de la unidad de disco se iluminará y usted oirá unos ruidos mientras que la computadora "lee" del disco. Luego usted debe ver en su pantalla lo siguiente :

```
La fecha actual es Mar 1-01-1980
Escriba la nueva fecha  (dd-mm-aa):
```

MS-DOS le pide que suministre la fecha.

1. Escriba la fecha. Por ejemplo, si la fecha es 6 de Julio de 1986, escriba el siguiente comando, luego presione la tecla ENTRAR :.

   06-07-86

   Si la fecha ya está correcta o usted no desea responder a este mensaje, presione la tecla ENTRAR para continuar al paso siguiente.

2. Escriba la hora de acuerdo a un reloj de 24 horas. Por ejemplo si es la 1:30 p.m., escriba lo siguiente y luego presione la tecla ENTRAR :

   13:30

   Si la hora ya está correcta o no quiere responder a este mensaje, presione la tecla ENTRAR.

MS-DOS no acepta el comando hasta que usted presione la tecla ENTRAR.

NOTA : Si comete un error cuanto está escribiendo la fecha o la hora, simplemente retroceda sobre el error y vuelva a escribir. Mientras que utiliza la tecla RETROCESO, notará que los caracteres desaparecen. Si comete un error y ya ha presionado la tecla ENTRAR presione las teclas CONTROL-ALTERNO-ELIMINAR simultáneamente para volver a iniciar el MS-DOS de nuevo.

Su pantalla se debe ver parecida a esto (la hora y la fecha pueden ser diferentes, dependiendo de lo que escribió en los pasos 1 y 2):

```
Fecha actual es Mar 1-01-1980
Escriba la nueva fecha (dd-mm-aa) : 06-07-86
La hora actual es 0:00:45:10
Escriba la nueva hora: 13:30

MS-DOS (R) Versión 3.30 por Microsoft (R)
 (s) Copyright Microsoft Corp 1981-1987

A>_
```

En este ejemplo, la unidad predeterminada es la unidad A, así que A> es la señal estandar de MS-DOS. Cuando vea la señal A>, MS-DOS está listo para escribir instrucciones.

COMO SALIR DE MS-DOS.

No hay un comando para salir de MS-DOS, pero usted puede terminar la sesión fácilmente siguiendo los pasos a continuación:

1. Asegúrese que el último comando se haya terminado de ejecutar. Usted debe ver el símbolo de MS-DOS (generalmente A>) en la pantalla.

2. Retire los discos flexibles de las unidades; colóquelos nuevamente en sus cubiertas protectoras y guárdelos en un lugar seguro, lejos del polvo, la humedad y objetos magnéticos.

3. Apague la computadora.

4. Apague el monitor.

III. ARCHIVOS Y DIRECTORIOS.

CONTROL DE ARCHIVOS.

Además de directorios, MS-DOS utiliza un área en el disco llamada Tabla de Asignación de Archivos (FAT). Cuando usted transfiere formato a un disco con el comando format (el cual se verá mas adelante), MS-DOS, copia esta tabla en el disco y crea un directorio vacío, llamado directorio raíz. En cada uno de sus discos, los directorios almacenan los archivos, y la Tabla de Asignación de Archivos (FAT) lleva un registro de la ubicación de cada archivo. La tabla también asigna el espacio libre que hay en los discos para que usted tenga espacio suficiente para crear nuevos archivos.

DIRECTORIOS DE NIVELES MULTIPLES.

Cuando hay más de un usuario para su computadora o cuando usted está trabajando en diferentes proyectos, el número de

archivos puede llegar a ser grande y difícil de manejar. Para organizar una gran cantidad de archivos, le puede ser conveniente mantener sus archivos separados de los de sus compañeros de trabajo, o puede querer organizar sus programas en categorías más convenientes.

Los directorios le permiten agrupar sus archivos en categorías convenientes. Cualquier directorio puede contener un máximo de 255 archivos. Estos directorios también pueden contener otros directorios (llamados subdirectorios). Esta organización de la estructura de los archivos se llama sistema de directorios de niveles múltiples o directorios jerárquicos.

NOTA: El número máximo de archivos o directorios que puede almacenar en el directorio raíz varía según el tipo de disco y unidad de disco que usted utiliza. Por lo general, el máximo es 112 para disco flexibles de 5 1/4 pulgadas, de dos caras y doble densidad. El número máximo de entradas en el directorio raíz de un disco de 3 1/2 pulgadas con capacidad de 1.44 megabytes de 224. Esta capacidad máxima para un directorio raíz también puede variar según la forma en que se dió formato al disco. No hay límite en el número de subdirectorios de un disco.

EL DIRECTORIO RAIZ.

El primer nivel en un directorio de niveles múltiples es el directorio raíz, el cual se crea automáticamente cuando se da formato a un disco y se comienza a colocar archivos en él. Usted puede crear directorios y subdirectorios adicionales dentro del directorio raíz.

EL DIRECTORIO DE TRABAJO.

El directorio en el que usted está trabajando se llama el directorio de trabajo. Los nombres de archivos y comandos explicados en este capítulo se relacionan con su directorio de trabajo y no se aplican a otros directorios de la estructura. Cuando enciende su computadora, usted comienza en el directorio de trabajo. Asimismo, cuando crea un archivo, lo crea en el directorio de trabajo.

Ya que puede colocar archivos en directorios diferentes, usted y sus compañeros de trabajo pueden tener archivos con los mismos nombres, pero con contenido distinto.

RUTAS DE ACCESO.

Una ruta de acceso al archivos es una secuencia de nombres de directorios seguidos por un nombre de archivo. Cada nombre de directorio está separado del anterior por el símbolo \. Una ruta de acceso es diferente que una ruta de acceso al archivo, ya que la primera no incluye un nombre de archivo.

El formato general de ruta de acceso al archivo es como sigue:

[\nombre de directorio][\nombre de directorio...]\nombre de archivo

Una ruta de acceso al archivo puede contener muchos nombres de directorio hasta con un máximo de 63 caracteres en total. Si la ruta de acceso al archivo comienza con el símbolo \, MS-DOS busca el archivo comenzando en la raíz del sistema de directorios. De otra manera, empieza en el directorio de trabajo y busca desde allí a lo largo de la ruta de acceso.

COMODINES.

Si está utilizando directorios de niveles múltiples, usted encontrará más fácil buscar archivos en sus discos utilizando dos caracteres especiales, llamados comodines. Los caracteres comodines son el asterisco (*) y el signo de interrogación (?). Ellos son útiles en las líneas de comando de MS-DOS ya que le dan flexibilidad cuando está especificando rutas de acceso y archivos.

EL COMODIN "?"

El signo de interrogación (?) en un nombre de archivo o en una extensión del nombre de archivo significa que cualquier carácter puede ocupar esa posición. El comando a continuación, por ejemplo, muestra todos los nombres de archivos en la unidad predeterminada que empiezan con los caracteres memo, que tienen cualquier carácter en la siguiente posición, que terminan con los caracteres ago y que tienen la extensión .txt :

```
dir memo?ago.txt
```

He aquí unos ejemplos de archivos que pueden ser listados por el comando anterior :

```
MEMO2AGO.TXT
MEMO9AGO.TXT
MEMOBAGO.TXT
```

EL COMODIN "*"

Un asterisco (*) incluido en un nombre de archivo o en una extensión al nombre de archivo, significa que cualquier carácter puede ocupar esa posición o cualquiera de las posiciones restantes en el nombre de archivo o extensión. Por ejemplo, el siguiente comando muestra todos los archivos en el directorio de la unidad predeterminada cuyos nombres comienzan con los caracteres memo y que tienen una extensión de .txt :

```
dir memo*.txt
```

IV. COMANDOS DE MS-DOS.

Existen dos tipos de comandos en MS-DOS :

- Comandos **internos**
- Comandos **externos**

COMANDOS INTERNOS.

Los comandos internos son los comandos más sencillos y más comúnmente utilizados. Cuando usted muestra el contenido del directorio de su disco de MS-DOS, no puede ver estos comandos porque forman parte de un archivo denominado command.com. Cuando escribe comandos internos, MS-DOS los ejecuta inmediatamente. Esto se debe a que fueron cargados dentro de la memoria de su computadora cuando se arrancó MS-DOS. Estos son los comandos internos de MS-DOS :

break	del	mkdir	set
chcp	dir	path	shift
chdir	echo	pause	time
cls	exit	prompt	type
copy	for	rem	ver
ctty	goto	ren	verify
date	if	rmdir	vol

COMANDOS EXTERNOS

Cualquier nombre de archivo con una extensión .com,.exe o .bat se considera un comandos externo. Por ejemplo, archivos tales como format.exe y diskcopy.exe son comandos externos. Como estos comandos son también archivos, usted puede crear nuevos comandos y agregarlos a MS-DOS. Los programas que crea con la mayoría de los lenguajes serán archivos ejecutables (.exe).

Cuando se utiliza un comando externo, no necesita escribir la extensión del nombre del archivo.

Nota: Si tiene más de un comando externo con el mismo nombre, MS-DOS ejecutará solamente uno de ellos, de acuerdo con el siguiente orden de precedencia : .com, .exe, .bat .

Los comandos externos de MS-DOS son los siguientes :

append	fdisk	recover
assign	find	replace
attrib	format	restore
backup	graftabl	select
chkdsk	graphics	share
command	join	sort
comp	keyb	subst
diskcomp	label	sys
diskcopy	mode	tree
exe2bin	more	xcopy
fastopen	nlsfunc	
fc	print	

Por lo general, MS-DOS recibe información de entrada desde el teclado y envía información de salida a la pantalla. Usted puede, sin embargo, redirigir este flujo de comandos de entrad y salida. Por ejemplo, puede desear que la entrada provenga de un archivo en lugar de venir del teclado y puede desear que los resultados de un comando vayan a un archivo o a una impresora en lugar de ir a la pantalla. Con la dirección también puede crear secuencias que permiten que la información de salida de un comando se convierta en la información de entrada para otro comando.

COMO REDIRIGIR LA INFORMACION DE SALIDA.

Para predeterminación, la mayoría de los comandos envían información de salida al monitor. En cambio, si usted desea enviarla a un archivo, debe de utilizar el signo mayor que (>) en el comandos. Por ejemplo, el siguiente comando muestra en la pantalla una lista del contenido del directorio en la unidad predeterminada :

```
dir
```

El comando dir pude enviar esta información a un archivo llamado contenid si escribe lo siguiente :

```
dir > contenid
```

Si el archivo contenid no existe, MS-DOS lo crea y almacena el lista del directorio en él. Si contenid ya existe, MS-DOS remplaza lo que ya está en el archivo por los nuevos datos.

Si desea agregar información a su directorio o agregar un archivo a otro (en lugar de remplazar el archivo completo), puede utilizar dos signos mayor que (>>) para indicarle a MS-DOS que agregue los resultados del comando (tal como un lista del directorio) al final de un archivo específico. Por ejemplo, el siguiente comando agrega un listado del directorio a un archivo ya existente llamado contenid :

```
dir >> contenid
```

COMO REDIRIGIR LA INFORMACION DE ENTRADA.

Frecuentemente es útil hacer que la información de entrada para un comando, venga de un archivo en lugar del teclado. Esto es posible en MS-DOS utilizando el signo menor que (<) en el comando. Por ejemplo, el siguiente comando ordena el archivo nombres y envía los resultados ordenados a un archivo llamado nomblist:

```
sort < nombres > nomblist
```

DESCRIPCION GENERAL DE LOS COMANDOS DE MS-DOS.

A continuación se da una breve descripción de todos los comandos de MS-DOS, posteriormente se dará una descripción más amplia de aquellos comandos que son más utilizados.

COMANDO	DESCRIPCION
APPEND	Establece una ruta de búsqueda para archivos de datos.
ASSING	Asigna una letra de la unidad a una unidad diferente.
ATTRIB	Define o muestra los atributos de archivo.
BACKUP	Realiza copias de respaldo de unos archivos de un disco a otro.
BREAK	Establece la comprobación de CONTROL-C.
CHCP	Muestra o cambia la tabla de códigos de trabajo para el procesador de comandos command.com
CHKDSK	Examina el directorio de la unidad predeterminada o designada y verifica su estado.
CLS	Borra la pantalla.
COMMAND	Procesa comandos internos de MS-DOS.
COMP	Compara el contenido de dos grupos de archivos.
COPY	Copia el archivo o los archivos especificados.

CTTY	Le permite cambiar el dispositivo desde el cual provienen los comandos.
DATE	Muestra y modifica la fecha.
DEL	Borra el (los) archivo(s) especificado(s) (erase)
DIR	Muestra el contenido del directorio especificado.
DISKCOMP	Compara discos.
DISKCOPY	Copia discos.
EXE2BIN	Convierte los archivos ejecutables (.exe) a formato binario.
EXIT	Sale del procesador de comandos y vuelve al nivel anterior.
FASTOPEN	Disminuye la cantidad de tiempo requerido para abrir archivos y directorios que se utilizan a menudo.
FC	Compara las diferencias entre dos archivos o grupos de archivos.
FDISK	Configura discos fijos para MS-DOS.
FIND	Busca una cadena de texto constante.
FORMAT	Da formato a un disco para recibir archivos de MS-DOS.
GRAFTABL	Carga una tabla de caracteres para gráficas.
GRAPHICS	Prepara a MS-DOS para imprimir gráficas.
JOIN	Asocia una unidad de disco a una ruta de acceso.
KEYB**XX**	Carga un programa de conversión del teclado.
LABEL	Graba etiquetas en los discos.
MKDIR	Crea un directorio (MD).
MODE	Define modos de operación para dispositivos.

MORE	Muestra información en el monitor una pantalla a la vez.
NLSFUNC	Carga información específica de un país.
PATH	Establece una ruta de búsqueda de comandos.
PROMPT	Le permite cambiar la señal de MS-DOS.
RECOVER	Recupera un disco o archivo defectuoso.
REN	Cambia el nombre del primer archivo por el del segundo archivo.
REPLACE	Remplaza las versiones anteriores de los archivos.
RESTORE	Restaura archivos respaldados.
RMDIR	Elimina un directorio (rd).
SELECT	Instala MS-DOS en un nuevo disco flexible con la información deseada pertinente a un país específico y el teclado asociado.
SET	Cambia un valor de cadena a otro en el ambiente o muestra el ambiente.
SHARE	Instala rutinas para compartir y proteger de archivos en red local.
SORT	Organiza datos ascendente o descendentemente.
SUBST	Substituye una cadena por una ruta de acceso a un archivo.
SYS	Transfiere archivos del sistema de MS-DOS desde una unidad a la unidad especificada.
TIME	Muestra y modifica la hora.
TREE	Muestra directorios y nombres de archivos.
TYPE	Muestra el contenido de un archivo.
VER	Muestra la versión de MS-DOS.
VERIFY	Verifica todas las escrituras en el disco.
VOL	Muestra la etiqueta del volumen.

DESCRIPCION DETALLADA DE ALGUNOS COMANDOS DE MS-DOS.

A continuación se muestran algunos comandos de MS-DOS explicados de una forma más detallada y se acompañan de sus principales calificadores, además de un ejemplo para su clara comprensión. Enseguida del comando se le pondra una (E) si es un comando externo y una (I) si es un comando interno.

BAKCUP (E) : Hace una copia de respaldo de uno o más archivos de un disco a otro disco.

Sintaxis :

bakcup [unidad1:][ruta de acceso][nombre de archivo][unidad2:][nombre de archivo]

donde :

Unidad1 : es la unidad de disco que contiene la información que desea copiar.

Unidad2 : es la unidad destino en donde guardará las copias de respaldo.

Indicador	Propósito
/s	Hace también copia de respaldo de subdirectorios.

EJEMPLO :

Suponga que Emilia quiere hacer copias de respaldo de todos los archivos incluidos en el directorio \usuarios\emilia de la unidad C a un disco en blanco y con formato que se encuentra en la unidad A. Para hacer esto, debe escribir lo siguiente:

```
backup c:\usuarios\emilia a:
```

CHDIR (I) : Cambia el directorio de trabajo a otra ruta de acceso; muestra el directorio de trabajo.

COMENTARIOS: Puede utilizar la abreviatura cd para el comando chdir.

Escriba cd \ para volver al directorio "raiz".

EJEMPLO :

Si utiliza chdir sin especificar una ruta de acceso, usted puede visualizar en pantalla el nombre del directorio de trabajo. Por ejemplo, si el directorio de trabajo es \usuarios\pero en la unidad B, y escriba el comando chdir y luego presiona la tecla ENTRAR, MS-DOS mostrará lo siguiente :

b:\usuarios\Pedro

Cuando se está trabajando en un sistema de directorios de niveles múltiples, para regresarse al directorio anterior ó "padre" lo podemos hacer de la siguiente forma :

cd ..

Con esto suponiendo que nos encontramos de un subdirectorio llamado planes de un directorio llamado proyectos, nos regresaríamos al directorio proyectos.

CHKDSK (E) : Examina el disco en la unidad especificada y busca errores.

El comando chkdsk acepta los siguientes indicadores :

Indicador	Propósito
/f	Corrige errores en el disco. Si no especifica este indicador, chkdsk no corrige los errores que encuentre en su directorio, sin embargo muestra mensajes acerca de los archivos que necesitan ser reparados.
/v	Muestra mensajes mientras se va ejecutando.

EJEMPLO :

Si quiere guardar el informe de estado chkdsk para utilizacion futura, puede redirigir la información de salida de chkdsk a un archivo llamado estado escribiendo lo siguiente :

chkdsk a: >estado

CLS (I) : Borra la pantalla de la computadora, dejando sólo la señal de MS-DOS y el cursor.

EJEMPLO :

Le puede resultar más cómodo trabajar con una pantalla "limpia". Si desea comenzar un nuevo proceso con una pantalla limpia, escriba :

```
cls
```

COPY (I) : Copia uno o más archivos a otro lugar. Este comando agrega un archivo a otro y copia archivos en el mismo disco.

Sintaxis :

copy [unidad:]ruta de acceso al archivo1 [unidad:][ruta de acceso al archivo2][/v][/a][/b]

Para agregar un archivo a otro :

copy ruta de acceso al archivo1 + ruta de acceso al archivo2 [...] ruta de acceso al archivoN archivo de resultado

Ejemplo :

Para copiar el archivo de la unidad a: ubicado en el subdirectorio trabajo, llamado text.dat a la unidad b: en el subdirectorio final, será con el siguiente comando :

```
copy a:\trabajo\text.dat b:\final\*.*
```

Para anexar el archivo nuevo.dat y el archivo viejo.dat y dejarlo en el archivo final.dat, será con el siguiente comando :

```
copy nuevo.dat+viejo.dat final.dat
```

DEL (I) : Elimina todos los archivos especificados por la unidad y la ruta de acceso al archivo.

Comentarios :

Este comando tiene un sinónimo llamado erase.

Ejemplos:

Para borrar el archivo inf.tex que está en la unidad b: estando nosotros en la unidad a, se logra mediante el comando :

```
del b:inf.tex
```

Supongamos que queremos borrar todos lo archivos que se encuentran en el subdirectorio llamado juegos que está abajo de la raíz de la unidad a:

```
del a:\juegos\*.*
```

DIR (I) : Muestra los archivos almacenados en el directorio

Sintaxis:

dir [unidad:][ruta de acceso al archivo][/p][/w]

Indicador	Propósito
/p	Selecciona el modo de página por página, haciendo una pausa después de mostrar cada página (pantalla) de información del directorio. Para continuar mostrando más información, presione cualquier tecla.
/w	Selecciona la visualización amplia que hace que MS-DOS muestre únicamente los nombres de los archivos y ninguna otra información referehte a los archivos. La visualización amplia lista hasta cinco archivos por línea.

Ejemplo:

Supongamos que queremos visualizar todos los archivos que estan en el subdirectorio llamado archivos de la unidad a:, esto se logra con el comando :

```
dir a:\archivos
```

DISKCOPY (E) : Copia el contenido del disco flexible de la unidad origen a un disco flexible con o sin formato en la unidad destino.

Sintaxis :

diskcopy [unidad1:] [unidad2:]

donde : unidad1 es la unidad de origen.
unidad2 es la unidad destino.

Comentarios :

Si el disco destino no tiene formato, **diskcopy** le da formato , con el mismo número de lados y sectores por pista que el disco origen.

Ejemplo :

Para copiar lo que contiene el disco de la unidad b: a la unidad a: estando nosotros desde la unidad a:, lo logramos con el siguiente comando :

diskcopy b: a:

FORMAT (E) : Dar formato al disco en la unidad especificada para aceptar archivos de MS-DOS.

Sintaxis :

format unidad:[1][/4][/8][/n:xx][/t:yy][/v][/s]

Comentarios :

El comando **format** inicializa el directorio y las Tablas de Asignación de Archivos en el disco. Usted debe utilizar este comando para darle formato a todos los disco nuevos antes de que MS-DOS pueda utilizarlos.

Ejemplo :

Para darle formato a un disco de 5 1/4 pulgadas y que además tenga el command.com y nos pida una etiqueta de volumen se logra con el siguiente comando :

format a: /s/v

KEYB (E) : Carga un programa de control del teclado.

Sintaxis :

keyb [xx[,[yyy],[[unidad:][ruta de acceso]nombre de archivo]]]

donde :

xx es un código de país de dos dígitos.

yyy es la tabla de códigos que define el juego de caracteres.

nombre de archivo es el nombre del archivo de control del teclado.

Comentarios :

xx es uno de los siguientes códigos de dos letras :

Código	Tipo de Teclado	Comando
us	Estado Unidos	keyb us
la	América Latina	keyb la
sp	España	keyb sp
gr	Alemania	keyb gr

MKDIR (I) : Crear un nuevo directorio.

Sintaxis :

mkdir [unidad:]trayecto

Comentarios :

Este comando tiene un sinónimo llamado md.

Ejemplo:

Para crear un subdirectorio llamado textos, en el disco de la unidad a:, es con el siguiente comando :

mkdir a:\textos

PATH (I) : Asigna una ruta de búsqueda de comandos.

Sintaxis :

path [unidad:][ruta de acceso][;[unidad:][ruta de acceso]...]

Comentarios :

El comando path le permite indicar a MS-DOS en qué directorios debe buscar los comandos externos -después que los busca en el directorio de trabajo. El valor predeterminado es ninguna ruta de acceso.

Ejemplo :

El siguiente comando le indica a MS-DOS que busque comandos externos en tres directorios :

path \usr\pedro;b:\usr\emilia;\bin

PRINT (E) : Imprime un archivo de texto en una impresora de líneas mientras que está procesando otros comandos de MS-DOS.

Sintaxis:

print [/d:dispositivo][/b:tamaño][unidad:][ruta de acceso al archivo]

Ejemplo :

Los siguientes dos comandos muestran cómo retirar el archivo lapiz.tst de la cola y luego agregar pluma.tst a la misma:

```
print lapiz.tst /c

print lapiz.tst /p
```

PROMPT (I) : Cambia la señal del sistema de MS-DOS.

Sintaxis :

prompt [[texto][$caracter]...]

A continuación se da una tabla para que se escriba enseguida

del comando prompt para obtener diferentes señales.

Escriba estos caracteres	Para obtener esta señal
$q	El carácter =
$p	El directorio de trabajo de la unidad predeterminada
$g	El carácter >
$v	El número de la versión

Ejemplo :

El siguiente comando establece la señal de la unidad como unidad:directorio de trabajo:

```
prompt $p
```

REN (I) : Cambia el nombre de un archivo

Sintaxis :

ren [unidad:][ruta de acceso]nombre de archivo1 nombre de archivo2

donde :

nombre de archivo1 es el nombre existente.

nombre de archivo2 es el nombre nuevo.

Comentarios :

Este comando tiene un sinónimo llamado rename

Ejemplo :

El siguiente comando, cambia el nombre de un archivo llamado cap10 (en la unidad B) a la parte10 :

```
ren b:cap10 parte10
```

RMDIR (I) : Elimina un directorio de la estructura de directorios de niveles múltiples :

Sintaxis :
rmdir [unidad:]ruta de acceso

Comentarios :

Rmdir elimina un directorio que está completamente vacío excepto por los símbolos "." y "..". Estos dos símbolos se refieren al directorio mismo y al directorio inmediatamente superior a éste, respectivamente. Antes de que pueda eliminar completamente un directorio, debe borrar los archivos y subdirectorios internos del directorio que desea eliminar.

Ejemplo :

Para eliminar el subdirectorio llamado pedro que está previamente vacío se logra con el siguiente comando :

```
rmdir \pedro
```

TYPE (I) : Muestra el contenido de un archivo de texto en la pantalla.

Sintaxis :

type [unidad:]nombre de archivo

Ejemplo :

Para ver el contenido del archivo llamado leyes.dat que está en el disco de la unidad a:, estando nosotros en la unidad c:, lo logramos con el siguiente comando :

```
type a:leyes.dat
```

V. PREPARANDO MS-DOS

Existen dos archivos especiales de MS-DOS, llamados AUTOEXEC.BAT Y CONFIG.SYS, estos archivos especiales de MS-DOS, le ayudarán a sacar mayor provecho del sistema operativo en la ejecución de comandos y programas de aplicación, y para el uso de diferentes dispositivos. Además, estos archivos especiales le ahorran tiempo, realizando tareas automáticamente cada vez que usted inicia MS-DOS.

EL ARCHIVO CONFIG.SYS

Cuando usted inicia MS-DOS, el sistema operativo busca automáticamente un archivo denominado config.sys en su disco del sistema. Ese archivo contiene comandos especiales que le permiten prepara (configurar) MS-DOS para ser utilizado conjuntamente con dispositivos o programas de aplicación.

Usted puede utilizar el comando dir para ver si el archivo config.sys ya está en su disco de MS-DOS. Si no se encuentra allí, puede utilizar Edlin para crearlo; si ya se encuentra en el disco del sistema, puede utilizar el comando type para mostrarlo en pantalla, o Edlin para modificarlo.

Ejemplo :

Aunque el archivo config.sys debe contener los siguientes comandos, no se preocupe si contiene más de estos comandos :

```
buffers=20

files=20
```

El comando buffers=20 determina el número de buffers, o bloques de memoria, que MS-DOS utilizar para almacenar datos. Si su sistema de directorios es muy grande, puede que desee definir un número mayor de buffers, por ejemplo 30.

El segundo comando en el archivo config.sys es files=20. Este comando determina el número de archivos que MS-DOS puede tener abiertos simultáneamente. Los programas tales como hojas de cálculo y bases de datos requieren que varios archivos estén abiertos al mismo tiempo. Si usted no define un valor para files en su archivo CONFIG.SYS, MS-DOS utiliza el valor de 8, el cual no sería suficiente para un programa grande tal como una base de datos.

EL ARCHIVO AUTOEXEC.BAT

MS-DOS también busca un segundo archivo cada vez que usted inicia su computadora. Este archivo se llama autoexec.bat y realiza una serie de comandos que usted normalmente ejecutaría al iniciar MS-DOS. Por ejemplo, puede utilizar este archivo con el objeto de preparar MS-DOS para el uso de un programa de aplicación específico.

Si existe un archivo autoexec.bat en el disco cuando usted inicia MS-DOS, MS-DOS no le pedirá automáticamente la hora y la fecha al comienzo de una sesión de trabajo con la computadora. Por

lo tanto, a menos que usted tenga un reloj instalado en su computadora, es importante poner los comandos de hora y fecha en su archivo autoexec.bat . De esta manera, MS-DOS le pedirá la hora y la fecha, y mantendrá actualizada la información en sus directorios en disco.

Ejemplo :

Para una computadora que tiene una unidad de disco flexible y una unidad de disco fijo. Puede contener las siguientes líneas:

```
date
time
path=c:;a:
prompt $p$g
dir
```

DIFERENCIAS ENTRE ESTOS ARCHIVOS ESPECIALES.

MS-DOS utiliza los archivos config.sys y autoexec.bat de manera distinta porque cada uno realiza diferentes tipos de comandos. Mientras que el archivo autoexec.bat puede contener cualquier comandos de MS-DOS o cualquier comando para iniciar un programa, el archivo config.sys sólo puede incluir ciertos comandos específicos de configuración.

Además, hay que volver a iniciar MS-DOS para ejecutar los comandos del archivo config.sys, para ejecutar los comandos en el archivo autoexec.bat, sólo hay que escribir la palabra autoexec.

VI. ARCHIVOS DE PROCESAMIENTO POR LOTES.

Usted puede encontrarse con frecuencia escribiendo repetidamente la misma secuencia de comandos para realizar algunas tareas comunes. Con MS-DOS puede colocar esta secuencia de comandos en un archivo especial llamado un archivo de procesamiento por lotes y luego ejecutar la secuencia completa de comandos simplemente escribiendo el nombre de este archivo. Note que no es necesario escribir la extensión del archivo de procesamiento aunque todos sus archivos por lotes sí deben incluir la extensión .bat en sus nombres, es decir todos los archivos de procesamientos por lotes deben de tener extensión .bat, independientemente del nombre

que se le quieran dar; pero para ejecutar un archivo de procesamiento por lotes no es necesario ejecutar la extensión del archivo.

Ejemplo :

El siguiente archivo es un archivo de procesamiento por lotes (ver que tiene comentarios que indica lo que hace) :

```
rem   Este es un archivo para dar formato
rem   y verificar nuevos discos
rem   Se llama vernuevo.bat
pause Inserte el disco nuevo en la unidad B:
format b: /v
chkdsk b:
```

Es decir al teclear a la señal de MS-DOS :

```
A> vernuevo
```

Lo que se hará es lo que indican los comandos que están especificados en el archivo de procesamiento por lotes "vernuevo.bat", que son :

1) Muestra en pantalla el mensaje de :

```
Este es un archivo para dar formato
y verificar nuevos discos
Se llama vernuevo.bat
```

2) A continuación se pide que se inserte un disco nuevo en la unidad B.

3) Posteriormente se da formato al disco que se introdujo en la unidad b, pidiendo la etiqueta de volumen.

4) En seguida se checa, el disco que se ha formateado.

Ejemplo :

El siguiente archivo de procesamiento por lotes lo que hace es listar los archivos que están en el directorio llamado usuarios de el disco duro, y luego se despliega un mensaje indicando que presione cualquier tecla para continuar y por último borra la pantalla.

```
dir c:\usuarios
pause
cls
```

Es importante hacer saber que cuando se está ejecutando un archivo de procesamiento por lotes cada uno de los comandos que estén especificados en el archivo serán mostrados en la pantalla de la computadora; pero existe una forma de que no aparezcan en la pantalla cuando se ejecutan y es antecediendo al comando el símbolo "@" que hace que se apague el "eco" del comando en la pantalla de la computadora.

Ejemplo :

Pruebe el siguiente archivo de procesamiento por lotes:

```
cd \
dir
```

Y ahora al mismo archivo, añadale a cada comando el símbolo "@" :

```
@cd \
@dir
```

Observe lo que sucede, ¿Qué diferencia hubo?

VII. EL EDITOR NORTON

Definición :

Un editor es un programa (Software) que nos permite la captura de información, nos brinda la posibilidad de generar nuestros archivos.

A pesar de que MS-DOS tiene su propio editor llamado EDLIN, existen muchos otros editores que también nos permiten hacer lo mismo, aquí no se explicará la forma de operar el editor EDLIN de MS-DOS, porque existen otros editores que son más sencillos de manipularse que el de MS-DOS.

Entre los muchos editores existentes hay uno en especial llamado : NORTON, este editor funciona de la siguiente manera :

Para poder editar un archivo a través de NORTON, es necesario que se tenga en el disco duro ó en disco flexible el archivo

llamado EDIT.COM, y una vez que se tenga este archivo lo que único que se tiene que hacer para poder editar un archivo es invocar al editor desde la unidad predeterminada donde se encuentra el archivo edit.com y enseguida el nombre del archivo que se quiere editar, es decir se teclea lo siguiente en la señal de MS-DOS.

```
C:\edit nuevo.txt
```

donde :

C	:	es la unidad predeterminada donde se encuentra el archivo edit.com
nuevo.txt	:	es el nombre del archivo que queremos editar.

Una vez hecho lo anterior, capturamos la información que deseamos y acontinuación para salvar esta información tecleamos F3 y a continuación la letra E (Exit), para no salvar nuestra información y salir de este editor tecleamos Q (Quit) después de F3.

Este editor tiene además una ayuda la cual la podemos invocar presionando la tecla F1, para que desaparezca la ayuda simplemente presionamos nuevamente la tecla F1.

Cabe hacer notar que existen muchos otros editores, pero aquí especialmente se está mostrando el editor de NORTON.

BIBLIOGRAFIA.

1) MS - DOS
REFERENCIA PARA EL USUARIO
COMPUBUR S.A. DE C.V.
SEPTIEMBRE 1988.

2) HOFFMAN, PAUL
SISTEMA OPERATIVO MS-DOS
GUIA DEL USUARIO
ED. McGRAW-HILL.

ULTRIX-32
Quick Reference Guide

Order No. AA-MF11A-TE

ULTRIX-32 Operating System, Version 3.0

Digital Equipment Corporation

This manual was written and produced by the ULTRIX Documentation Group in Nashua, New Hampshire.

Contents

This quick reference guide contains condensed information about the commands in Section 1 of the *ULTRIX-32 Reference Pages*.

The first section of the guide contains information about some of the more complicated commands in Section 1:

- adb
- dbx
- ed
- ex
- vi
- sh
- csh
- sccs
- mail
- nroff and troff
- ms
- mh

The second section of this guide condenses the rest of the Section 1 commands.

To use this guide effectively, you should be familiar with the commands. For thorough specifications, of course, refer to the Section 1 Reference Pages.

Details of Selected Commands 1

This section details some of the more complicated commands from Section 1 of the Reference Pages. These commands are:

- adb Debugger
- dbx Debugger
- ed Text Editor
- ex Text Editor
- vi Screen Editor
- sh Bourne Shell
- csh C Shell
- SCCS Commands
- mail Commands
- nroff and troff Primitives
- ms Macro Package
- mh Message Handler

adb Debugger

adb [- w] [- k] [- I*dir*] [*objfil* [*corfil*]]

Options

objfil — Normally an executable program file, preferably containing a symbol table. Default is a.out.

corfil — A core image file produced after objfil is executed.

- I*dir* — Specifies a directory where files to be read with $< or $<< will be sought. Default is /usr/lib/adb.
- k — Specifies kernel memory mapping. Should be used when core is a crash dump or /dev/mem.
- w — Creates objfil and corfil if necessary, and opens them for reading and writing.

Commands

!	Escape and execute shell command.
newline	Repeat last command.
>name	Set dot to the variable or register named.
?[*format*]	Print from object file.
?*[*format*]	Print from object file using second mapping triple.
/[*format*]	Print from core file.
/*[*format*]	Print from core file using second mapping triple.
=[*format*]	Print address.

Formats

o	Print 2 bytes in octal.
O	Print 4 bytes in octal.
q	Print signed octal.
Q	Print long signed octal.
d	Print short decimal.
D	Print long decimal.
x	Print 2 bytes in hex.
X	Print 4 bytes in hex.
u	Print as unsigned decimal.
U	Print as long unsigned decimal.
f	Print 32-bit floating point number.
F	Print double-floating point.
b	Print 1 byte in octal.
c	Print addressed character.
C	Print addressed character. Control characters are printed as ^X, the delete character as ^?.
s *n*	Print the addressed characters until a zero or *n* is reached.
S *n*	Print a string, *n* is length of string.
Y	Print 4 bytes in date format.
i *n*	Print as machine instruction.
a	Print value of dot in symbolic form. Types are:
	/ Local or global data symbol
	? Local or global text symbol
	= Local or global absolute symbol
p	Print addressed value in symbolic form.
t	Tabs to appropriate tab stop.
r	Print a space.
n	Print a newline.
"..."	Print the enclosed string.
^	Dot is decremented by the current increment, nothing is printed.
+	Dot is incremented by one. Nothing is printed.
-	Dot is decremented by one. Nothing is printed.
[?/]l *value mask*	Value mask words with *mask* until *value* found.
[?/]w *value*	Write *value* into addressed location.

[?/]m *b1 e1 f1*[?/]
Change map parameters.

Command

$*modifier*	Miscellaneous commands

Modifiers

<f	Read commands from file f.
>f	Append output to file f.
r	Print registers and instruction addressed by pc.
b	Print all breakpoints, counts, and commands.
c	Call stack backtrace.
d	Set default radix to address and report new value.
e	Print external variables.
w	Set page width to *address*. Default is 80.
s	Set limit on number of symbol matches to *address*. Default is 255.
o	Set default for integer input to octal.
q	Exit from adb.
v	Print all nonzero variables in octal.
m	Print the address map.
p	Set process context.
?	Print process ID.

Command

:*modifier*	Manage a subprocess

Modifiers

bc	Set breakpoint at *address*, *c* is command to execute.
d	Delete a breakpoint at *address*.

r	Run object file as a subprocess.
c*s*	Continue subprocess with signal *s*.
s*s*	Single step subprocess *s* number of times.
k	Kill subprocess.

Variables

0	Last value printed
1	Last offset part of an instruction source
2	Previous value of variable 1
9	Count on last $< or $<< command
b	Base address of data segment
d	Data segment size
e	Entry point
m	Magic number
s	Stack segment size
t	Text segment size

Expressions

.	The value of dot
+	Dot plus increment
^	Dot minus increment
"	Last address typed
integer	Decimal number, octal if it starts with 0o or 0O, decimal if it starts with 0t or 0T, hex if it starts with 0x or 0X.
integer.fraction	A 32-bit floating point number
'cccc'	The ASCII value of up to four characters
<name	The value of variable or register *name*
symbol	The value of *symbol*
_symbol	The value of an external *symbol*
routine.name	Address of variable *name* in *routine*
(exp)	The value of expression *exp*

Monadic operators

**exp*	Contents of location addressed by *exp* in *corfil*
@ exp	Contents of location addressed by *exp* in *objfil*
- exp	Integer negation
~exp	Bitwise complement
#exp	Logical negation

Dyadic operators

e1+e2	Integer addition
e1-e2	Integer subtraction
*e1*e2*	Integer multiplication
e1%e2	Integer division
e1&e2	Bitwise conjunction
e1|e2	Bitwise disjunction
e1#e2	*E1* rounded up to the next multiple of *e2*

dbx Debugger

dbx [- r] [- i] [- k] [- I*dir*] [- c*file*] [*objfile* [*coredump*]]

objfile An object file containing symbol information.

coredump A file containing a core dump.

Options

- r	Executes *objfile* immediately; prompts only on error.
- i	Assumes standard input is a terminal.
- k	Maps memory addresses for kernel debugging.
- I *dir*	Adds *dir* to directory search path.
- c *file*	Executes commands in *file* before reading from *stdin*.

Commands

Execution and Tracing Commands

run [*args*] [<*filename*] [>*filename*]
rerun [*args*] [<*filename*] [>*filename*]
Start executing *objfile*, passing *args* as command line arguments: < or > can be used to redirect input or output. Using **rerun** without arguments passes the previous argument list to the program.

trace [**in** *procedure/function*] [**if** *condition*]
trace *source-line-number* [**if** *condition*]
trace *procedure/function* [**in** *procedure/function*] [**if** *condition*]
trace *expression* **at** *source-line-number* [**if** *condition*]
trace *variable* [**in** *procedure/function*] [**if** *condition*]
Traces the expression or variable at the specified line or routine.

stop if *condition*
stop at *source-line-number* [**if** *condition*]
stop in *procedure/function* [**if** *condition*]
stop *variable* [**if** *condition*]
Stops execution when the given line is reached, procedure or function called, variable changed, or condition true.

status [> *filename*]
Displays active **trace** and **stop** commands.

delete *command-number...*
Removes all traces or stops corresponding to the given numbers. **Delete*** removes all existing breakpoints and tracepoints at once.

catch *number* or *signal-name*

ignore *number* or *signal-name*
Start or stop trapping a signal before it is sent to the program.

cont [*signal*]
Continues execution from where it stopped, passing *signal* if specified. Signal may be specified by name or by integer value.

step
Executes one source line, stepping into procedures.

next
Executes up to the next source line, stepping over procedures.

return [*procedure*]
Continues until return to *procedure* is executed, or until the current procedure returns if none is specified.

call *procedure(parameters)*
Executes the object code associated with the named procedure or function.

Printing Variables and Expressions

assign *variable = expression*
Assigns value of expression to variable.

dump [*procedure*] [>*filename*]
Prints names and values of variables in given procedure. If procedure given is ".", then all active variables are dumped.

print *expression* [, *expression...*]
Prints out values of the expressions.

whatis *name*
Prints declaration of the given name.

which *identifier*
Prints full qualification of the given identifier.

whereis *identifier*
Prints full qualification of all occurrences of the given identifier. (See *which*.)

where
Prints a list of active procedures and functions.

down [*count*]

up [*count*]
Moves the current scope up or down the stack *count* levels. Default is 1.

Accessing Source Files

/regular expression[/]
?regular expression[?]
Search forward (/) or backward (?) in current source file for given pattern.

edit [*filename*]

edit *procedure/function-name*
Invokes default editor on *filename* or current source file if unspecified.

file [*filename*]
Changes current source file name to *filename*.

func[*procedure/function*]
Changes current function.

list [*source-line-number* [,*source-line-number*]]

list *procedure/function*
Lists specified source lines or procedure. Default is 10 lines unless range is specified.

use *directory-list*
Sets directory search path for source files.

Command Aliases and Variables

alias *name name*

alias *name "string"*

alias *name (parameters) "string"*
Defines an alias for *name*. If no parameters are specified, all aliases are displayed.

set *name* [= *expression*]
Defines debugger variable.

The following have special meaning:

$frame
Specifies stack frame. Useful in kernel debugging.

$hexints
$hexchars
$hexoffsets
$hexstrings

Prints characters, integers, offsets from registers, or character pointers in hexadecimal.

$listwindow
Specifies number of lines to list around a function. Default is 10.

$mapaddrs
Starts or stops address mapping.

$unsafecall
$unsafeassign
Turns off typechecking for subroutine calls, system calls, or between two sides of an assignment statement.

unalias *name*
Removes alias.

unset *name*.
Deletes definition or alias of *name*.

Machine Level Commands

tracei [*address*] [if *cond*]
tracei [*variable*] [at *address*] [if *cond*]
stopi [*address*] [if *cond*]
stopi [at] [*address*] [if *cond*]

stepi
nexti
Trace, stop, or step by machine instruction rather than by source line.

address, address/ [*mode*]

address / [*count*] [*mode*]
Displays memory contents from first address up to second address or until count items are displayed. The following modes specify how memory is to be displayed:

b	Displays a byte in octal.
c	Displays a byte as a character.
d	Displays a short word in decimal.
D	Displays a long word in decimal.
f	Displays a single precision real number.
g	Displays a double precision real number.
i	Diplays machine instruction.
o	Diplays a short word in octal.
O	Displays a long word in octal.
s	Displays string of characters terminated by a null byte.
x	Displays a short word in hexadecimal.
X	Displays a long word in hexadecimal.

Miscellaneous Commands

help	Displays a synopsis of dbx commands.
quit	Exits dbx.
sh	*command-line* Passes command line to the shell for execution. The SHELL environment variable determines which shell is used.
source	*filename* Reads dbx commands from given *filename*.

ed Text Editor

ed [-] [- x] [file]
red [-] [- x] [file]

file	Name of file to be edited.
-	Suppresses display of character counts.
- x	Similates x command.

Commands

a	Append text.
c	Change text.
d	Delete text.
e *file*	Edit file.
E *file*	Edit file, but display no warning message.
f *file*	Change name of current file to *file*. Default is current file name.
(1,$)g/*RE*/*command list*	Execute commands globally on each line containing search expression (*RE*).
(1,$)G/*RE*/	Interactively execute commands globally on each line containing search expression (*RE*).
h	Displays short error message explaining reason for most recent ? diagnostic.
H	Displays error message for all subsequent ? diagnostics and previous ?.
i	Insert text.
j	Join lines.
kx	Mark line with character *x*.
l	List lines and give ASCII equivalents for nonprintable characters.
m*a*	Move addressed lines after line *a*.

n	Number and list each addressed line.
p	Print text lines.
P	Turns on and off *ed* prompt.
q	Quit edit session.
Q	Quit edit session and issue no warning message.
(**$**) **r** *file*	Read *file* after the addressed line.
s/*RE*/*rep*/	Substitute replacement expression (*rep*) for search expression (*RE*).
s/*RE*/*rep*/**g**	Substitute replacement expression (*rep*) for every occurrence of search expression (*RE*).
t *a*	Copy lines to address *a*.
u	Undo the previous substitution.
(1,$) **v**/*RE*/*command list*	Execute *command list* on all lines that do not contain search expression (*RE*).
(1,$) **V**/*RE*/	Execute interactive global command on all lines that do not contain search expression (*RE*).
(1,$) **w** *file*	Write lines to *file*.
X	Demands a key string from the standard input.
(**$**) **=**	Print current line number.
!*shell command*	Execute shell *cmd*.
(.+1) <newline>	Print next line.

ex Text Editor

ex [**-**] [**- v**] [**- t** tag] [**- r**] [**+***command*] [**-** name...
edit [ex options]

name Name of file to be edited.

Options

-	Suppress all interactive user feedback.
- v	Equivalent to *vi* command.
- t *tag*	Edit the file containing the *tag* and position the editor at its definition. Equivalent to an initial *tag* command.
- r	Recover named file after an editor or system crash. If no file is specified, display a list of saved files.
- l	Set the *showmatch* and *lisp* options.

Commands

a	Append text.
c	Change text.
co	Copy text lines.
d	Delete text lines.
e *file*	Edit *file*.
f	Print current file name.
g/*exp*/*cmds*/	Execute commands globally on all lines containing search expression (*exp*).
i	Insert text.
j	Join text lines.
l	List text lines and show tabs and end of lines.
ma *x*	Mark line with character argument *x*.

m *addr*	Move text lines after *addr*.
n	Edit next file.
nu	Number and list each addressed line.
o	Open new line for text insertion.
p	Print text lines.
pu	Put back deleted or yanked lines.
q	Quit edit session.
r *file*	Read *file* into buffer.
set	Set or list ex options.
sh	Execute shell.
s/*exp*/*rep*/	Substitute replacement expression (*rep*) for search expression (*exp*).
u	Undo last editing command.
vi	Enter vi (full-screen display) mode.
w *file*	Write buffer back to *file*.
x	Exit edit session and write to file, if necessary.
y	Yank specified lines to buffer.
! *cmd*	Exit and execute the specified command (*cmd*).

vi Screen Editor

vi [- t tag] [- r] [+*command*] [- l] [- w*n*] name ...

Screen Control Commands

<CTRL/L>	Reprints current screen.
<CTRL/R>	Reprints current screen and eliminates @ lines.
z<RETURN>	Moves current line to top of screen.
z-	Moves current line to bottom of screen.
z.	Moves current line to center of screen.
/*pattern*/z-	Moves line containing *pattern* to bottom of screen.
z*n*.	Sets screen size to *n* lines.
<CTRL/Y>	Exposes one more line at top of screen.
<CTRL/E>	Exposes one more line at bottom of screen.
n<CTRL/E>	Exposes *n* more lines at bottom of screen.
n<CTRL/Y>	Exposes *n* more lines at top of screen.

Paging Commands

<CTRL/F>	Pages forward one screen.
<CTRL/B>	Pages back one screen.
<CTRL/D>	Pages down half screen.
<CTRL/U>	Pages up half screen.

Cursor Positioning Commands

j	Moves cursor down one line, same column.
*n*j	Moves cursor down *n* lines, same column.

k	Moves cursor up one line, same column.
*n*k	Moves cursor up *n* lines, same column.
h	Moves cursor back one character.
*n*h	Moves cursor back *n* number of characters.
l	Moves cursor forward one character.
*n*l	Moves cursor forward *n* number of characters.
+	Moves cursor to beginning of next line.
<RETURN>	Moves cursor to beginning of next line.
-	Moves cursor to beginning of previous line.
^	Moves cursor back to first nonblank space on current line.
0	Moves cursor to beginning of current line.
$	Moves cursor to end of current line.
<SPACE>	Moves cursor forward one character.
n|	Moves cursor to column *n*.
w	Moves cursor forward one word.
*n*w	Moves cursor forward *n* number of words.
b	Moves cursor back one word.
*n*b	Moves cursor back *n* number of words.
e	Moves cursor to end of current word. When repeated, moves cursor to end of next word.

H	Moves cursor to beginning of first line on screen.
L	Moves cursor to beginning of last line on screen.
M	Moves cursor to beginning of middle line on screen.
*n*G	Moves cursor to beginning of line *n*. Default is last line of file.
f*x*	Moves cursor forward on current line to next occurrence of *x*.
F*x*	Moves cursor back on current line to previous occurrence of *x*.
t*x*	Moves cursor forward on current line to character before *x*.
T*x*	Moves cursor back on current line to character before *x*.
;	Repeats previous f, F, t, or T command.
,	Reverses direction of f, F, t, or T command.
/*pattern*	Moves cursor forward to next occurrence of *pattern*.
?*pattern*	Moves cursor back to previous occurrence of *pattern*.
n	Repeats last / or ? pattern search.
N	Reverses direction of last / or ? pattern search.
%	Finds matching ( ), { }, or [ ] if cursor on either one of the pair.
''	Moves cursor to previous context. Functional only

	after altering text or searching a pattern.
''	Moves cursor to beginning of previous context line. Functional only after altering text or searching a pattern.
m*x*	Marks current position with letter *x* (*a-z*).
'*x*	Moves cursor to position previously marked *x*.
'*x*	Moves cursor to beginning of line containing position marked *x*.
]]	Moves cursor to next section (for text containing formatting macros).
[[	Moves cursor to previous section (for text containing formatting macros).
)	Moves cursor to beginning of next sentence.
}	Moves cursor to beginning of next paragraph.
(	Moves cursor back to previous sentence.
}	Moves cursor back to previous paragraph.

Text Insertion Commands

a	Appends text after cursor until stopped by pressing the escape key.
A	Appends text at end of current line until stopped by pressing the escape key. Same as $a.
i	Inserts text before cursor until stopped by pressing the escape key.

I	Inserts text at beginning of current line until stopped by pressing the escape key. Same as ^i.
o	Opens new line below current line for text insertion until stopped by pressing the escape key.
O	Opens new line above current line for text insertion until stopped by pressing the escape key.
<CTRL/D>	Backs out text one shift width. Auto indent must be set.
<CTRL/0D>	Backs out text to left edge of screen. Auto indent must be set.
<CTRL/H>	Overwrites last character during text insertion.
<DELETE>	Overwrites last character during text insertion.
<CTRL/T>	Indents text one shift width. Shift width must be defined.
<CTRL/W>	Deletes previous word during text insertion.
<ESC>	Stops text insertion.
<CTRL/V>	Inserts and displays following control character (next argument).

Text Deletion Commands

dw or dW	Deletes current word.
x	Deletes current character.
X	Deletes previous character.
*n*x	Deletes *n* characters.
dw or dW	Deletes current word.

*n*dw	Deletes *n* words. Default is current word (or remainder of current word).
de	Deletes current word but leaves punctuation.
dd	Deletes current line.
*n*dd	Deletes *n* lines.
*n*dj	Deletes current line plus next *n* lines.
D	Deletes from cursor to end of line.
d/*pattern*	Deletes all text up to *pattern*.
df*x*	Deletes text through the given *x*.

Text Change Commands

Change commands work with objects, put you into insert mode, shift text to right or left to fit, and must be ended with the escape key.

cc	Changes characters on current line until stopped with escape key.
*n*cc	Changes *n* (number) of lines.
cw or cW	Changes characters of current word until stopped with escape key.
*n*cw	Changes characters of next *n* words.
c$	Changes text up to the end of the line.
ct*x*	Changes text up to the given letter *x*.
C	Changes remaining text on current line until stopped by pressing the escape key.
	Changes case of current character.

xp	Transposes current and following characters.
J	Joins current line with next line.
>>	Moves current line one shift width to the right.
<<	Moves current line one shift width to the left.
>L	Moves all lines between cursor and end of screen one shift width to the right.
<L	Moves all lines between cursor and end of screen one shift width to the left.
n<< or *n*>>	Moves *n* number of text lines one shift width to left or right respectively.

Text Replacement Commands

Replacement commands work on a single character or to the end of the line, put you into overwrite mode, do not shift any text, and must be ended with the escape key.

r*x*	Replaces current character with *x*.
R	Replaces existing text on current line until stopped with the escape key.

Text Substitution Commands

Subsitution commands work on a character or line and can be repeated, put you into insert mode, shift text from right or left to fit, and must be ended with the escape key.

s	Substitutes text for current character until stopped by pressing the escape key.

S	Substitutes text for current line until stopped by pressing the escape key.
:s	Substitutes new word(s) for old. Written as: :<addr range> s/old/new/g, where addr range is a range of line numbers. Example: :1,$ s/car/truck/g.
&	Repeats last substitution (:s) command.

Undo and Redo Commands

These commands apply to all text alteration activities.

u	Undoes last change made.
U	Restores current line.
.	Repeats last change.

Buffer Usage Commands

*n*Y	Yanks *n* lines to unnamed buffer. Default is current line.
yy	Yanks current line to unnamed buffer.
*n*yy	Yanks *n* lines to unnamed buffer. Default is current line.
"*n*p	Puts back text from *n*th previous delete (1-9).
yw	Yanks current word to unnamed buffer.
y*n*w	Yanks *n* words to unnamed buffer.
p	Puts yanked text line(s) after current line or yanked words after cursor.
P	Puts yanked text line(s) before current line or yanked words before cursor.

"*x*n*Y*	Yanks *n* lines to buffer *x [a-z]*. Default is current line.
"*x*y*L*	Yanks text from cursor to end of screen into buffer *x[a-z]*.
"*xy/pattern*	Yanks text from cursor to *pattern* into buffer *x[a-z]*.
"*x*n*dd*	Deletes *n* lines and saves them in buffer *x[a-z]*. Default is current line.
"*x*p	Puts text from buffer *x[a–z]* before current line.
"*x*P	Puts text from buffer *x[a–z]* after current line.

You may also yank, copy, or delete a vi command into a named buffer and then execute the command with @*x*, where *x* is any letter *a-z* that you have used to name the buffer in which you have put your command.

File Manipulation Commands

:w *file*	Writes changes to *file*. Default is current file.
:wq	Writes changes to current file and quits edit session.
:w! *file*	Overwrites *file*. Default is to overwrite current file.
:q	Quits edit session (no changes made).
:q!	Quits edit session and discards changes.
:e *file*	Edits *file*.
:e!	Discards changes and reedits current file.
:e +*n file*	Edits *file* and places cursor at line *n*. Default is to place cursor at end of *file*.
:e#	Edits alternate file. Allows shifting of text between files by using named buffers.
:sh	Executes shell and then returns to edit session.
:!*cmd*	Escapes to execute *cmd* and then returns to edit session.
:n	dits next file in argument list.
:f *name*	Changes name of current file to *name*. Default is to print name of file and current line number.
<CTRL/G>	Displays current file name and line number.
:r *file*	Reads contents of file into current file at current cursor position.
:so *file*	Sources given ex or vi command file.
:set nu	Numbers each text line.

Command Filters

A command filter takes input from the buffer, modifies it, and inserts its output back into the buffer in place of the input text.

!x**command**	Uses text of cursor positioning command *x* buffer as standard input to the given command and replaces the buffer text with output from the command. Example: **!Gsort** will sort all text to the end of the file, then replace those lines with the sorted output.
!!**command**	Performs the command on the current line, then replaces the current line with the output from the command. Example: **4!!sort** will sort the next 4 lines and replace those lines with the sorted output.
:r !**command**	Reads in the output of the command after the current line. Example: **:r !date** will add the date after the current line.
:w !**command**	Sends contents of the buffer to a command without affecting them. Examples: **:w !wc** will count the buffer contents; **:w !!pr** will print the buffer contents;;**1,25w !!pr** will print lines 1 through 25.

Startup Commands

Map commands and named command buffers can be set up in the EXINIT variable located in the .login file in your home directory. This creates your editing environment whenever vi is invoked. Unused keys available for vi command mapping are K, V, g, q, v, *, =. It is also possible to redefine any built-in vi command keys, as shown in the following example (E, s and S).

Example .login file EXINIT setting:

```
setenv EXINIT 'set ai aw ic sw=4 redraw terse|map g
G| map v ~~~~ |map E :e#<CTRL/M>|map s
eas<esc>| map K G:r \spell % <CTRL/M> |map S
G1/\<escA\>esc0"ad$dd@ a<CTRL/M>'
```

ai	Auto indent
aw	Auto write
ic	Ignore case on searches
sw	Shift width = 4 spaces (<, >, CTRL/D, CTRL/T).
redraw	Redraw screen after deletes.
terse	Terse error messages.
wm	Warp margin (spaces from right edge of screen). Automatic line splitting.
ws	Wrap scan around end of buffer on searches.
map	Define your own commands.
:map*x* *command*	Map operator where *x* is any letter, *command* is any vi/ex command or combination of commands. Examples: :map g G causes g command to act like G; :map v ~~~~ causes letter v to change the case of the next 4 letters.

Source Command Files

Editor options and key mappings that only affect the current invocation of vi can be specified in a file that you source after invoking vi.

Examples:

A file named .dialexrc containing commands "set noredraw slow" would be useful on a dial-in terminal.

A file named .textexrc containing these commands could be used when editing English text:

```
set ic wm=10
ab U ULTRIX
ab UEG ULTRIX Engineering Group
```

Invoke this command from within vi as follows:

```
:so ~/.textexrc
```

Bourne Shell Format

sh [- ceiknrstuvx] [arg] ...

arg	Name of each command sequence to be executed. If not specified, invokes sh command for interactive execution.
- c *string*	Executes the specified commands.
- e	Exits immediately on unsuccessful execution of a command.
- i	Invokes sh for interactive session.
- k	Places all specified keywords into current execution environment.
- n	Reads all commands but does not execute them.
- s	Reads stdin until CTRL/D and uses this input as the command sequences to be executed.
- t	Exit after executing one command only.
- u	Treats all undefined variables as an error.
- v	Displays each input line upon execution.
- x	Displays all command sequences before execution.
-	Turns off the **- x** and **- v** options.

Command Execution

cmd1 | *cmd2*	Pipes output of *cmd1* as input to *cmd2*.
cmd1 ; *cmd2*	Executes commands in succession.
cmd &	Executes *cmd* in background.
cmd1 && *cmd2*	Executes *cmd2* only if *cmd1* succeeds.
cmd1 || *cmd2*	Executes *cmd2* only if *cmd1* fails.

Input/Output Redirection

> *file*	Redirects standard output to *file*.
[*n*]> *file*	Redirects file descriptor *n* to *file* output.
< *file*	Redirects standard input from *file*.
[*n*]< *file*	Redirects input from *file* to any file descriptor *n*.

>> *file*	Redirects standard output and appends to *file*.
[*n*]>> *file*	Redirects file descriptor *n* and appends to *file*.
<< *word*	Reads standard input until encounters line containing only single *word*.
>& *n*	Duplicates file descriptor *n* and uses as standard output.
[*m*]>& *n*	Duplicates file descriptor *n*, appends data from file *m* to *n* and uses as standard output.
<& *n*	Duplicates file descriptor *n* and uses as standard input.
[*m*]<& *n*	Duplicates file descriptor *n*, appends file descriptor *m* to *n* and uses as standard input.
<&-	Closes standard input.
>&-	Closes standard output.

File Name Generation

?	Matches any single character.
*	Matches any string of characters, including a null string.
[*xyz*]	Matches any of the enclosed characters (*xyz*). Initial dots are never matched.

Variable Substitution

$*x*	Substitutes defined value for *x*.
$*	Substitutes all arguments for *.
$@	Same as $* but quotes each argument.
$-	Substitutes option flags used to invoke current shell.
$#	Substitutes number of specified arguments for #.
$?	Substitutes exit status of last command for ?.
$$	Substitutes process ID of current shell for $.

$!	Substitutes process ID of last background command for !.
$HOME	Substitutes path of default home directory for HOME.
$PATH	Substitutes default execution search paths for PATH.
$MAIL	Substitutes path of your mail file for MAIL.
$PS1	Substitutes primary prompt string for PS1.
$PS2	Substitutes secondary prompt string for PS2.
$IFS	Substitutes interfield separators for IFS.
$TERM	Substitutes terminal type for TERM.
$n	Parameter at position n.
${p-w}	Substitutes value for *parameter* if set; otherwise substitutes *word*.
${p=w}	Substitutes value of parameter that is set to *word*.
${p?w}	Substitutes value of set *parameter*. If not set, prints *word* and exits from shell. Prints standard message if *word* is omitted.
${p+w}	Substitutes *word* for *parameter* if set.

Quoting

x	Quotes specified character *x*.
'xxx'	Quotes specified character string *xxx*, including special characters: \ ', ", $.
"xxx"	Quotes specified character string *xxx*, excluding \ ', ", $.

The use of double quotation marks (" ") cause suppression of file name generation, while permitting the variable expression to be used. Single quotation marks (' ') causes suppression of file name generation and variable substitution.

Control Flow Constructs

All of the following control flow constructs must be the first command in the list in order to be recognized.

for *name* [**in** *word...*] **do** *list* **done**

case *word* **in** [*pat1* [*pat2*]...] *list*;;... **esac**

if *list* **then** *list* [**elif** *list* **then** *list*]...
[**else** *list*] **fi**

while *list* [**do** *list*] **done**

(*list*)	Executes commands in *list* in a subshell.
{*list*}	Executes commands in *list* in current shell.
:	Serves as null command to identify line as comment.
. *file*	Reads in commands to be executed from specified *file*.
break [*n*]	Exits from loop at specified level *n*.
continue [*n*]	Continues with next iteration of loop at specified level *n*.
cd [*arg*]	Changes current directory to *arg*. The shell parameter $HOME is the default *arg*.
exit [*n*]	Exits with specified status *n*.
eval [*arg...*]	Reads and executes each specified argument.
exec [*arg...*]	Executes each specified argument without creating a new process. If no arguments are given, shell file descriptors are modified with the use of input/output directional symbols.
export [*name...*]	Exports each named variable to current execution environment.
login [*arg...*]	Equivalent to 'exec login arg ...'.
read *name*	Reads standard input and assigns *name*.
readonly [*name...*]	Marks each named variable readonly.
set [- eknptuvx [*arg...*]]	Sets the following /bin/sh shell options:

- e Exits immediately if a command fails when not interactive.
- k Places all keyword arguments in the environment for a command, not just those that precede the command name.
- n Reads commands but does not execute them.
- t Exits after reading and executing one command.
- u Treats unset variables as an error when substituting.
- v Prints shell input lines as they are read.
- x Prints commands and their arguments as they are executed.
- Turns off the - x and - v options.

shift Shifts positional parameters left: $1 = $2 and so on.

times Prints accumulated process times.

trap [*arg*] [*n*] Executes *arg* only if signal *n* is received.

umask [*nnn*] Sets the user file creation mask to the octal value *nnn*. Default: Prints the current value of the mask.

wait [*n*] Waits for specified process (*n*). Default is to wait for all child processes.

C Shell Format

csh [- cefinstvVxX] [arg ...]

arg Name of each command to be executed and command options and arguments, if any.

- V Displays verbose information as input is read.
- X Displays command sequence before execution.
- c Reads commands from specified file (next argument).
- e Exits if command terminates abnormally.
- f Invoked with fast start up: does not read .cshrc.
- i Invokes interactive shell (default).
- n Reads commands, but does not execute them.
- s Reads input from stdin.
- t Exits after one command.
- v Displays input as read (verbose mode).
- x Prints on execution.

Command Execution

cmd1 | *cmd2*
Pipes output of *cmd1* to input of *cmd2*.

cmd1 ; *cmd2*
Executes commands in succession.

cmd & Executes *cmd* in background.

cmd1 && *cmd2*
Executes *cmd2* only if *cmd1* succeeds. Exit status is 0.

cmd1 || *cmd2*
Executes *cmd2* only if *cmd1* fails. Exit status is 1.

x Quotes specified character *x*.

'*xxx*' Quotes specified character string *xxx*, including special characters: \,",$.

"*xxx*" Quotes specified character string *xxx*, excluding \ ' " $.

(*cmd*) Executes *cmd* in a subshell.

Input/Output Redirection

> *file*	Redirects standard output to *file*.
< *file*	Redirects standard input from *file*.
>> *file*	Redirects standard output and appends it to *file*.
<< *word*	Reads standard input until it encounters any line containing only *word*.

Input/Output Modifiers

!	Overrides noclobber.
&	Redirects standard error output also to specified *file*.

File Name Generation

?	Matches any single character.
*	Matches any string of characters, including a null string.
[*xyz*]	Matches any of the enclosed characters (*xyz*). There is an error if no match occurs.
~*name*	Matches specified user's (*name*) home directory. Default is to match your home directory.
{*x*,*y*,*z*}	Matches any of the enclosed characters, preserving specified order, without errors.

Variable Substitution

$*name*	Substitutes the value of a variable name for $*name*.
$*name*[*n*]	Substitutes the value of the *nth* member of $*name*.
$*name*[*m*-*n*]	Substitutes the values of *mth* through *nth* members of $*name*.
$#*name*	Substitutes the number of words in the value for the variable for $#*name*.
$0	Substitutes *name* of file from which command input is read for $0.

$*n*	Substitutes $argv[n] for $n.
$*	Substitutes all arguments to the shell for $*.
$?	Substitutes exit status of last command for $?.
$$	Substitutes process ID of parent shell for $$.

Predefined Variables

argv	Contains array of arguments to the shell.
cd*path*	Contains list of alternate directories to search for chdir command.
cwd	Contains path for current working directory.
echo	Causes each command and argument to be echoed just before execution with all translations.
history	Contains size of the history list.
home	Contains path for home directory.
ignoreeof	Disables logging out by CTRL/D.
mail	Contains path for mail file.
noclobber	Prevents accidental file destruction.
noglob	Inhibits file name expansion.
nonomatch	Prevents errors in nonmatching file expansions.
notify	Notifies when job completes.
path	Contains search paths for command execution.
prompt	Contains prompt string.
shell	Contains path to shell program.
status	Contains status return by last command.
term	Contains terminal type.
time	Controls printing of command timing
verbose	Echos commands only with history substitutions.

Control Flow Constructs

foreach *name* (*list*)

```
end

switch (string)
case str1:
    .
    .
    .
  breaksw
default:
    .
    .
    .
  breaksw
endsw

if (expr) cmd  or  if (expr) then
    .
    .
    .
else if (expr2) then
    .
    .
    .
else
    .
    .
    .
endif

while (expr)
    .
    .
    .
end
```


Shell Commands

<CTRL/Z>	Stops current job.
alias [*name*] [*string*]	Sets string alias for name. Without the string, displays alias for name. Without both name and string, displays all aliases.
alloc	Displays current memory usage.
bg [%*n*]	Puts specified job (*n*) in backgound.
beak	Resumes execution after closest foreach or while statement.
cd [*name*]	Changes working directory to named directory.
continue	Continues execution of closest foreach or while statement.
dirs	Prints names in current directory stack.
echo [- n] [*list*]	Echoes each argument in specified list with newline at the end, unless - n.
eval *arg...*	Reads and executes each specified argument.
exec *cmd*	Executes *cmd* in place of current shell.
exit	Quits current shell.
fg [%*n*]	Puts specified job (*n*) in foreground.
glob *list*	Similar to echo, but no escapes or null delimiters
goto *string*	Transfers execution to *string*.
hashstat	Displays hash table statistics.
history	Displays history list.
jobs	Displays status of active jobs.
kill - sig job	Sends signal to the job. The default signal is SIGTERM. Kill - 1 displays the signals.
limit	Sets job limits.
login	Invokes /bin/login in place of login shell.
logout	Terminates login shell.

newgrp	Changes group identification.
nice	Alters execution priority.
nohup	Ignores hangups.
notify	Notifies of job completion.
onintr	Specifies interrupt handling.
popd [+*n*]	Removes top or *n*th entry from directory stack.
pushd	Exchanges top two elements on directory stack.
pushd *name*	Changes directory to *name* and pushes *name* on directory stack.
rehash	Recomputes command location hash table.
repeat *n cmd*	Repeats *cmd n* times.
set [*name*] [*string*]	Set shell variable (*name*) to *string*. Default is to display all variable settings.
setenv *name string*	Sets environment variable (*name*) to *string*.
shift	Shifts argv list left one position.
source *name*	Reads commands for current shell from *name*.
stop [%*n*]	Stops specified job (*n*). Default is to stop current job.
suspend	Stops current shell.
time [*cmd*]	Displays summary of execution times for *cmd*. Default is to display times for current shell.
umask *mode*	Turns off specified permission bits (*mode*) on all created files.
wait	Waits for background jobs.
%*n*	Restates specified stopped job (*n*).
@	Displays value of all shell variables.

SCCS Commands

sccs *command* [*command option*] [*file*] [sccs]

Commands

admin – Creates and administers SCCS files

- -a Adds user (-a*login*) to list of those permitted to make changes to SCCS files.
- -d Deletes specified SCCS flag (-d*flag*) from file.
- -e Erases user (-e*login*) from list of those permitted to make changes to SCCS files.
- -f Turns on specified flag (-f*flag*).
- -h Checks structure of SCCS file.
- -i Creates using specified file (-i*name*) as initial contents.
- -m Inserts specified modification request numbers (-m*rlist*) into SCCS file.
- -n Creates new SCCS file.
- -r Indicates initial delta release number (-r*SID*). Used only with -i keyletter. Default is 1.1.
- -t Replaces descriptive text with contents of specified file (-t*name*).
- -y Inserts specified text (-y*comment*) as initial comment.
- -z Rebuilds the SCCS file checksum.

Admin flags (used with f and d options):

b	Allows branches.
c*ceil*	Retrieves highest release by the get command for editing (must be a positive number no higher than 9999, the default number).
f*floor*	Retrieves lowest release by the get command for editing (must be a positive number between 0 and 9999). Default is 1.
d*SID*	Sets default delta version number (d*n*) for get or edit commands.

i Treats "no id keywords" as a fatal error.

j Allows multiple, concurrent updates to the same version of any SCCS file, using the get command for editing.

llist Specifies list of releases to which deltas can no longer be made. See admin(1) for syntax.

n Creates a null delta in any release that is skipped when a delta is made in a new release. Example: Making delta 5.1 after 2.7 skips releases 3 and 8. These are created but are null and serve as anchor points for branch deltas.

q*text* Substitutes user definable text for all occurrences of the %Q% keyword in SCCS file text retrieved by get.

m*mod* Substitutes module name of the SCCS file for all occurrences of the %M% keyword in file text retrieved by get. Default is name of the SCCS file with the leading s. removed.

v[*pgm*] Prompts for modification request (MR) numbers as the reason for creating a delta.

t*type* Module type.

cdc - Changes delta commentary

- m[*mrlist*] Adds or deletes specified modification numbers (- m*list*). Must be used with - r option.
- r*SID* Indicates delta version number.
- y[*comment*] Replaces comments already existing for the delta specified by the - r keyletter.

check - Displays information only about files being edited and returns exit status. (Similar to info.)

clean - Removes recreatables files.

comb - Combines deltas.

- - c Preserves specified deltas (- c*list*).
- - o Accesses reconstructed files at release of delta.
- - p Oldest delta to preserve (- p*SID*).
- - s Generates script that reports: file name, size (after), size (before), and percentage changed.

create - Creates SCCS file.

deledit - Produces a delta and gets new g-file for edit.

delget - Produces a delta and gets new g-file

delta

- - g Specifies list of deltas to be ignored - g[*list*].
- - m Indicates modification request number (- m[*mrlist*]).
- - n Does not delete edited file.
- - p Prints differences before and after deltas are applied.
- - r Indicates new delta release number (- r*SID*).
- - s Suppresses messages.
- - y Creates delta with specified commentary (- y[*comment*]).

edit - Get SCCS file for editing

- - b Creates a branch
- - i Includes specified list of deltas (- i*list*).
- - r Gets specified version (- r*SID*) for editing.
- - x Excludes specified list of deltas (- x*list*).

fix - Removes delta and edits earlier version

get - Get copies of SCCS files

- - a Retrieves specified delta sequence number (- a*seq-no.*).
- - b Gets delta from branch. Used with - e.
- - c Does not apply deltas created after specified date-time (- c*cutoff* in the form YY[MM[DD[HH[MM[SS]]]]]).

- e Gets specified delta version for editing. Equal to edit.
- g Suppresses getting text from SCCS file.
- i Includes specified list of deltas (- i*list*).
- k Does not expand ID keywords.
- l Writes delta summary to l- file.
- m Precedes each line with delta version number.
- n Precedes each line with identification keyword.
- p Writes text to stdout.
- r Gets specified version number (- r*SID*).
- s Suppresses all messages.
- t Gets most recent (top) delta.
- x Excludes specified list of deltas (- x*list*).

info - Displays information about files being edited

- b Ignores branches.
- u Displays information about files edited by named user (- u*name*).

prs - Displays information from SCCS files.

- a Displays information for both existing and removed deltas.
- d Displays information specified by dataspec (- d*dataspec*).
- e Displays information for all deltas created before and including specified delta (- e*SID*).
- l Displays information for all deltas created including and after specified delta (- l*SID*).
- r Indicates delta version number (- r*n*).
- t Prints descriptive text.

prt - Display changes made to SCCS files.

rmdel - Removes deltas

- r Removes specified version (- r*SID*).

sccsdiff - Display differences between SCCS files.

- p Pipes output through pr command.
- r Specifies first delta (- r*SID1*).
- r Specifies second delta (- r*SID2*).

- s Specifies segment size for bdiff (- s*n*).

sccshelp - Displays help for SCCS error messages.

tell - Displays only names of SCCS files being edited.

unedit - Undo SCCS edit command.

unget - Undo SCCS get command.

- n Retains copy of SCCS file.
- r Indicates delta version number (- r*SID*).
- s Suppresses all messages.

val - Validates SCCS files.

- m Compares specified value (- m*name*) with %M% keyword.
- r Indicates delta version number (- r*SID*).
- s Suppresses all error messages.
- y Compares specified type (- y*type*) with %Y% keyword.

what - Displays SCCS ID keywords in object files.

mail program

mail [- v] [- i] [- n] [- s subject] [user]

mail [- v] [- i] [- n] - f [name]

mail [- v] [- i] [- n] - u user

- f	Checks mbox or the specified file instead of your normal account for mail.
- i	Ignores all terminal generated interrupt signals.
- n	Ignores /usr/lib/Mail.rc during startup.
- s subject	Uses specified subject for mail header.
- u user	Checks specified user's account for mail.
- v	Displays verbose delivery information.

Write and Send Commands

a	Sets or lists aliases.
alt	Lists addresses where messages are not to be sent.
d*n*	Deletes one message or range of messages. (For example: d1-10 deletes messages 1 through 10. Default is to delete current message.
dp or dt	Deletes current messages and prints next message.
e	Edits messages with ex editor, unless the .mailrc file set *editor* command specifies another editor.
ex or x	Exits from mail, with no modifications
m	Sends mail to current message sender. Use m *name* to specify another.
q	Quits mail session.
R	Routes reply to originator of message only. Does not reply to other recipients of the original message.
r or respond	Routes reply to sender and all who received message.
se	Sets options in form of "option=value" or "option". If no argument if given, displays all variable values.
u	Undeletes all messages marked for deletion by the d command.
u*n*	Undeletes numbered message.
unalias	Takes list of names defined by alias commands and cancels the list of users.
unset	Discards values assigned to option. The inverse of set.
v	Edits messages with vi editor.
!*command*	Escapes to a shell, then executes the given shell command.

Read Commands

f*n* or from	Displays numbered message header. f displays current message header, f* displays all message headers.
h	Lists current range of headers, in message group determined by the speed of your terminal. Terminals with a baud rate of < 1200 list 5 headers, 10 if the baud rate is 1200, and 20 if the baud rate is > 1200. The default range can be overridden by the screen command. h*n* begins with message *n*.
help	Displays summary of mail commands. Identical to ? command.
ignore	Eliminates selected header items on displayed message. When used by itself, ignore displays the current list of ignored header fields.
n	Displays next message.
p	Displays messages on terminal. Identical to t.
P	Displays messages on terminal plus extensive header information. Identical to T.

size	Lists size, in characters, of current message in mail directory. size* gives size of all messages.
so	Reads *mail* commands from a file.
t	Displays messages on terminal. Identical to p.
T	Displays messages on terminal plus extensive header information. Identical to P.
top*m*	Displays *n* number of top lines of message *m*, as determined by the variable *toplines*. Default is 5.
z	Displays windows of message headers. See h and screen commands for window sizes. z or z+ moves forward to next window; z- moves to previous window.
+	Displays next message in sequence.
-	Returns to previous message.
?	Displays summary of mail commands.
<RETURN>	Displays next message in sequence.

Store Commands

ch	Changes directory to that specified. Default is login directory.
co *n file*	Copies message *n* to *file* and does not delete that message from mail list.
ho *n*	Holds message *n* in system mail box. ho* holds all messages.
fi or file	Switches to new mail file or folder. Identical to folder command.
folder	Switches to new mail file or folder. With no argument, tells you which file you are currently reading. With argument given, writes changes made to current file and reads in new file. Conventions used are: #, previous file; %, system mailbox; %user, user's system mailbox; &, your 7mbox file; +folder, a file in your folder directory.

folders	Lists names of folders in folder directory.
mbox	Sends message to *mbox* file after you quit mail session.
pre	Preserves message in system mailbox. Identical to ho command.
s *file*	Saves messages in a named file.
sh	Invoke /bin/sh shell.
w *file*	Writes message to a given file. Identical to s command.

Tilde Escape Commands

Tilde escapes are only recognized at the beginning of lines and are used to perform special functions when composing messages.

~b *name*	Sends blind carbon copy to specified user *name*. (Name will not be included in cc: list).
~c *name*	Send carbon copy to specified user (*name*).
~d	Reads mail from dead.letter file.
~e	Edits message with ex editor.
~f *message*	Reads *message* into message being sent.
~h	Edits the message header.
~m *message*	Reads *message* into message being sent, shifted one tab space to the right.
~p	Prints current message.
~q	Quits current message, but saves contents in dead.letter file. (Be sure that nosave is not set in .mailrc file.)
~r *filename*	Reads *file* named into the message.
~s *string*	Substitutes *string* for current subject field.
~t *name*	Adds specified user (*name*) to direct recipient list.
~v	Edits message with vi editor.
~w *filename*	Writes message to *file*.

~|*cmd*	Pipes message to *cmd* as a filter.
~~*string*	Inserts text string in the message prefaced by a single ~. If the escape character has been changed, it must be doubled in order to send the message.
~!*cmd*	Executes specified command (*cmd*).
~:	Executes mail commands. (Example: ~:10 prints out message number 10).
~?	Displays brief summary of tilde commands.

Binary Variables

These options are controlled with the **set** and **unset** commands. Check first to see whether or not they are set.

append	Appends messages to end of *mbox* file.
ask	Prompts for subject header.
askcc	Prompts for carbon copy recipients.
autoprint	Causes **d** command to act like **dt** command.
dot	Interprets a period alone on a line as the terminator of the message is sent.
hold	Causes messages to be held in system mailbox by default.
ignore	Ignores interrupts from terminal, such as CTRL/C and CTRL/D, when set ignore = 1.
ignoreeof	Refuses to accept CTRL/D as the end of a message.
metoo	Includes self if in distribution list.
msgprompt	Prompts for message text and indicates how to terminate the message, when sending mail.
nosave	Prevents the copying of an aborted message to the dead.letter file.
save	Saves interrupted messages in dead.letter file.

quiet	Does not print mail version.
verbose	Puts *mail* into verbose mode, displaying message on the terminal when you send a message.

String Variables

EDITOR	Contains path for default text editor (~e).
SHELL	Contains path for shell.
VISUAL	Contains path for vi editor (~v).
crt	Determines how long a message must be before *more* is used to read it.
escape	Contains default escape character.
folder	Contains name of directory to use for storing folders of messages. If name begins with /, it is considered to be an absolute pathname: otherwise, the directory is found relative to the home directory.
record	Contains path for file in which all outgoing mail is saved.
screen	Defines window size for the **h** and **z** commands. Example: **set screen = 18** causes mail to display message headers in groups of 18.
toplines	Contains number of lines to print for the top command.

nroff and troff Primitives

NOTE: Troff is not supported by Digital Equipment Corporation.

Font and Character Control Primitives

.ps *n* Sets point size to *n* (troff only).
.ft *x* Changes font to *x* (troff only).
.ul *n* If nroff, underlines next *n* input lines. If troff, italicizes next *n* input lines.

Page Control Primitives

.pl +- *n* Sets page length to *n*. Default measure is *n* lines.
.bp +- *n* Begins new page and numbers page *n*.
.pn +- *n* Numbers next page *n*.
.ne *n* Begins new page if *n* vertical space does not fit on current page. Default measure is *n* lines.
.po +- *n* Sets left margin (page offset) to +- *n*. Default measure is *n* ems.

Text Filling, Adjusting, Centering Primitives

.br Forces break in text.
.nf Does not fill or adjust text.
.fi Begins text filling.
.na Does not adjust text (ragged right margin).
.ad *x* Adjusts text with mode *x*.
.ce *n* Centers next *n* input lines.

Vertical Spacing Primitives

.ls *n* Sets line spacing: places *n*- 1 blank lines between output text lines.
.sp +- *n* Sets vertical spacing: negative *n* spaces backwards. Default measure is *n* lines.

Line Length and Indenting Primitives

.ll +- *n* Sets line length to *n*. Default measure is *n* lines.

.in +- *n* Indents text by *n*. Default measure is *n* ems.
.ti +- *n* Indents next output line *n*. Default measure is *n* ems.

Macros, Strings, Number Registers Primitives

.de *xx* Defines macro *xx*. (Characters .. on separate line ends definition.)
.ds *xx string* Defines specified string (*xx*) to *string*.
Calls contents of one-character string *x*.
Calls contents of two-character string *xx*.
.nr *r xx* Sets specified value (*xx*) to number register *r*.
n*x* Calls contents of one-character register *x*.
n*xx* Calls contents of two-character register *xx*.

Conditional Primitives

.if *x cmds* Executes specified commands (*cmds*), if condition *x* is true.
.if !*c cmds* Executes specified commands (*cmds*), if condition *c* is false.
.if *n cmds* Executes specified commands (*cmds*), if *n* > 0.
.if !*n cmds* Executes specified commands (*cmds*), if *n* <= 0.
.if *'str1'str2' cmdsstr1'str2' cmds*'u>(231u+1n) .br Executes specified commands (*cmds*), if str1 is identical to str2.
.if !*'str1'str2' cmdsstr1'str2' cmds*'u>(231u+1n) .br Executes specified commands (*cmds*), if str1 is not identical to str2.
.ie *c cmds* Constitutes "if" portion of "if...else" statement. Similar if statements above.
.el *cmds* Constitutes "else" portion of "if...else" statement

Input/Output Options

These options may appear in any order but must be used in the general form:

nroff *options files*.

- -o*list* Prints only pages whose page numbers appear in the comma-separated *list* of numbers and ranges. Range *N-M* means pages *N* through *M*; an initial *-N* means from the beginning of page *N*; and a final *N-* means from *N* to the end.
- -n*N* Number the first generated page *N*.
- -s*N* Stop printer every *N* pages to allow paper loading or changing, then resumes printing upon receipt of a newline. Default is 1.
- -m*name* Prepend macro file */usr/lib/tmac/tmac.name* to the input *files*.
- -r*aN* Set register *a* (one-character) to *N*.
- -i Read standard input after the input files are exhausted.
- -q Invoke the simultaneous input-output mode of the rd request.
- -T*name* Prepare output for the specified terminal (*name*).
- -e Produce equally spaced words in adjusted lines, using full terminal resolution.
- -h Use output tabs during horizontal spacing to speed output and reduce output character count. Tabs are assumed to be 8 nominal character widths.

- ms Macro Package

NOTE: Troff is not supported by Digital Equipment Corporation.

Format Macros

- .1C Specifies 1 column format.
- .2C Specifies 2 column format.

Paragraphs

- .PP Begins indented paragraph.
- .LP Begins left-adjusted paragraph.
- .IP *xx n* Begin indented paragraph: indents body *n* spaces and prints tag (*xx*) in margin.

Section Head Macros

- .NH Specifies numbered section heading.
- .SH Specifies unnumbered section heading.

Indent and Display Macros

- .RS Increases relative indent.
- .RE Ends relative indent.
- .DS *x* Begins displayed text: indented but not filled. Values for *x* are: L (left-adjusted text) or C (centered text).
- .DE Ends displayed text.

Keep Macros

- .KS Begins block of text to be kept together.
- .KF Begins floating keep: text kept together as a unit, but output only when there is adequate space.
- .KE Ends keeps.

Footnotes

- .FS Begins footnote text.
- .FE Ends footnote.

Font and Point Size Macros

- .I Begins italics (troff). Begins underlined text (nroff).

.B	Begins bold text (troff).
.R	Begins Roman type (default).
.SM	Changes to smaller point size (troff).
.LG	Changes to larger point size (troff).
.NL	Returns to normal point size (troff).
.UL	Underlines specified word.

eqn and tbl Preprocessors Macros

.EQ	Begins equation.
.EN	Ends equation.
.TS	Begins table.
.TS H	Begins table with repeated headings.
.TH	Ends heading section of table.
.TE	Ends table.

Macros for Papers

.DA	Forces date on each page.
.RP	Begins released paper format: cover sheet with title and abstract.
.TL	Specifies title on following line(s).
.AU	Specifies author(s) on following line(s).
.AI	Specifies author's institution on following lines.
.AB	Begins abstract.
.AE	Ends abstract.
.ND	Does not print date.

mh – Message Handler

Programs to send, receive, save, and retrieve messages.

ali – list mail aliases

Syntax

ali [-alias *aliasfile*] [-list] [-nolist] [-normalize] [-nonormalize] [-user <*useradr*>] [-nouser] *aliases* ... [-help]

anno – annotate messages

Syntax

anno [+*folder*] [*msgs*] [- -component *field*] [-inplace] [-noinplace] [-text body] [-help]

burst – explode digests into messages

Syntax

burst [+folder] [*msgs*] [-inplace] [-noinplace] [-quiet] [-noquiet] [-verbose] [-noverbose] [-help]

comp – compose a message

Syntax

comp [+*folder*] [*msg*] [-draftfolder +*folder*] [-draftmessage *msg*] [-nodraftfolder] [-editor *editorname*] [-noedit] [-file *filename*] [-form *formfile*] [-use] [-nouse] [-whatnowproc *program*] [-nowhatnowproc] [-help]

dist – redistribute a message to additional addresses

Syntax

dist [+*folder*] [*msg*] [-annotate] [-noannotate]
[-draftfolder +*folder*] [-draftmessage *msg*]
[-nodraftfolder] [-editor *editorname*] [-noedit]
[-form *formfile*] [-inplace] [-noinplace]
[-whatnowproc *program*] [-nowhatnowproc] [-help]

folder – set folder or display current foldername

Syntax

folder [+*foldername*] [*msg*] [-all] [-fast] [-nofast]
[-header] [-noheader] [-pack] [-nopack] [-recurse]
[-norecurse] [-total] [-nototal] [-print] [-noprint] [-list]
[-nolist] [-push] [-pop] [-help]

folders – list folders and contents

Syntax

folders [*folder*] [*msg*] [-fast] [-nofast] [-header]
[-noheader] [-pack] [-nopack] [-recurse] [-norecurse]
[-total] [-nototal] [-print] [-noprint] [-list] [-nolist]
[-push] [-pop] [-help]

forw – forward messages

Syntax

forw [+*folder*] [*msgs*] [-annotate] [-noannotate]
[-draftfolder FI+folder] [-draftmessage *msg*]
[-nodraftfolder] [-editor *editorname*] [-noedit]

[-filter *filterfile*] [-form *formfile*] [-format] [-noformat]
[-inplace] [-noinplace] [-whatnowproc *program*]
[-nowhatnowproc] [-digest *list*] [-issue *number*]
[-volume *number*] [-help]

inc – incorporate new mail

Syntax

inc [+*foldername*] [-audit *audit-file*] [-noaudit]
[-changecur] [-nochangecur] [-form *formatfile*]
[-format *string*] [-file *name*] [-silent] [-nosilent]
[-truncate] [-notruncate] [-width *columns*] [-help]

mark – mark messages

Syntax

mark [+*foldername*] [*msgs*] [-sequence *name*...] [-add]
[-delete] [-list] [-public] [-nopublic] [-zero] [-nozero]
[-help]

mh – Message Handler

mhl – produce formatted listings of MH messages

Syntax

/usr/new/lib/mh/mhl [-bell] [-nobell] [-clear] [-noclear]
[-folder +*foldername*] [-form *formfile*] [-length *lines*]
[-width *columns*] [-moreproc *program*] [-nomoreproc]
[files ...] [-help]

mhmail - send or read mail

Syntax

mhmail [*addrs* ...] [-body *text*] [-cc *addrs* ...]
[-from *addr*] [-subject *subject*] [-help]

mhpath - print full pathnames of MH messages and folders

Syntax

mhpath [+*foldername*] [*msgs*] [-help]

msgchk - check for messages

Syntax

msgchk [-nodate] [-notify *all/mail/nomail*] [users ...]
[-help]

msh - MH shell

Syntax

msh [-prompt *string*] [-scan] [-noscan] [-topcur]
[-notopcur] [file] [-help]

next - show the next message

Syntax

next [+*foldername*] [-header] [-noheader] [-showproc
program] [-noshowproc] [switches for showproc] [-help]

packf - compress a folder into a single file

Syntax

packf [+*folder*] [*msgs*] [-file *name*] [-help]

pick - select messages by content

Syntax

pick [+*folder*] [*msgs*] [-and ...] [-or ...] [-not ...]
[-lbrace ... -rbrace] [-search] [- -component *pattern*]
[-after *date*] [-before *date*] [-datefield *field*]
[-sequence *name* ...] [-public] [-nopublic] [-zero]
[-nozero] [-list] [-nolist] [-help]

prev - show the previous message

Syntax

prev [+*foldername*] [-header] [-noheader]
[-showproc *program*] [-noshowproc] [-switches for
showproc] [-help]

prompter - prompting editor front-end

Syntax

prompter [-erase *chr*] [-kill *chr*] [-prepend] [-noprepend]
[-rapid] [-norapid] file [-help]

rcvstore - incorporate new mail asynchronously

Syntax

rcvstore [+ folder] [- create] [- nocreate] [- sequence *name*] [- public] [- nopublic] [- zero] [- nozero] [- help]

refile - file message in other folders

Syntax

refile [*msgs*] [- draft] [- link] [- nolink] [- preserve] [- nopreserve] [- src + *foldername*] [- file *filename*] + *folder* [- help]

repl - reply to a message

Syntax

repl [+ *folder*] [*msg*] [- annotate] [- noannotate] [- cc *all/to/cc/me*] [- nocc *all/to/cc/me*] [- draftfolder + *foldername*] [- draftmessage *msg*] [- nodraftfolder] [- editor *editor*] [- noedit] [- fcc + *foldername*] [- filter *filterfile*] [- form *formfile*] format] [- noformat] [- inplace] [- noinplace] [- query] noquery] [- width *columns*] [- whatnowproc program] [- nowhatnowproc] [- help]

rmf - remove folder

Syntax

rmf [+ *foldername*] [- interactive] [- nointeractive] [- help]

rmm - remove messages

Syntax

rmm [+ *folder*] [*msgs*] [- help]

scan - produce a one-line-per-message scan listing

Syntax

scan [+ *folder*] [*msgs*] [- clear] [- noclear] [- form *formatfile*] [- format *string*] [- header] [- noheader] [- width *columns*] [- help]

send - send a message

Syntax

send [- alias *aliasfile*] [- draft] [- draftfolder + *foldername*] [- draftmessage *msg*] [- nodraftfolder] [- filter *filterfile*] [- nofilter] [- format] [- noformat] [- forward] [- noforward] [- msgid] [- nomsgid] [- push] [- nopush] [- verbose] [- noverbose] [- watch] [- nowatch] [- width *columns*] [file ...] [- help]

show - show (list) messages

Syntax

show [+ *folder*] [*msgs*] [- draft] [- header] [- noheader] [- showproc *program*] [- noshowproc] [switches for showproc] [- help]

slocal - MH receive-mail hooks

Syntax

slocal $HOME/.maildelivery [-form *formfile*]
[switches for *postproc*] address ... [-help]
/usr/new/lib/mh/rcvpack file [-help]
/usr/new/lib/mh/rcvtty [*command* ...] [-help]

sortm - sort messages

Syntax

sortm [+*folder*] [*msgs*] [-datefield *field*] [-verbose]
[-noverbose] [-help]

whatnow - prompting front-end for send

Syntax

whatnow [-draftfolder +*folder*] [-draftmessage *msg*]
[-nodraftfolder] [-editor *editorname*] [-noedit]
[-prompt *string*] [file] [-help]

whom - report to whom a message would go

Syntax

whom [-alias *aliasfile*] [-check] [-nocheck] [-draft]
[-draftfolder +*folder*] [-draftmessage *msg*]
[-nodraftfolder] [file] [-help]

Other Section 1 Commands 2

This section summarizes the remaining commands from Section 1 of the Reference Pages.

2780e - spooler for the IBM 2780 emulator

Syntax

2780e [-m] [-a] [-q] [-b] [-t] [-S*file*] [-#*num*] *file*...
[-o *file*...]

Options

-#	Waits for *num* files to be received as output from job and gives default file names in the form *Ruseridpid*.
-a	Send file as priority job.
-b	Transmits the file to an IBM system that accepts multiple record transmission.
-m	Notifies user by mail that file was sent and output was received.
-o	Name output files with specified file names.
-q	Prepares the file for transmission and places it in /usr/spool/rje but

does not call 2780d to transmit.

-S	Sends contents of file to the IBM system as a sign-on card.
-t	Sends data in transparent mode.

3780e – spooler for the IBM 3780 emulator

Syntax

3780e [-C] [-m] [-a] [-q] [-t[b]] [-S*file*] [-#*num*] *file*... [-o *file*]

Options

-#	Waits for *num* files to be received as output from job and gives default file names in the form *Ruseridpid*.
-a	Send file as priority job.
-b	Transmits the file to an IBM system that accepts multiple record transmission.
-C	Prevents the compression of spaces when files are sent.
-m	Notifies user by mail that file was sent and output was received.
-o	Name output files with specified file names.
-q	Prepares the file for transmission and places it in /usr/spool/rje but does not call 2780d to transmit.
-S	Sends contents of file to the IBM system as a sign-on card.
-t	Sends data in transparent mode.

-tb	Transmits the file to an IBM system that accepts multiple 80-column card records in transparent mode.

addbib – create or extend bibliographic database

Syntax

addbib [-p *promptfile*] [-a] *database*

Options

-a	Suppresses prompting for an abstract.
-p	Causes use of a new prompting skeleton, defined in *promptfile*.

apply – apply a command to a set of arguments

Syntax

apply [-ac] [-*n*] *command args*...

apropos – locate commands by keyword lookup

Syntax

apropos *keyword*...

ar – archive and library maintainer

Syntax

ar *-key [posname] afile name...*

Options

d	Deletes the named files from the archive file.
m	Moves the named files to the end of the archive.
p	Prints the named files in the archive.
q	Appends the named files to the end of the archive file.
r	Replaces the named files in the archive file.
t	Prints a table of contents of the archive file.
x	Extracts the named files.
a	Tells the ar command that new files should be placed after *posname*.
b	Tells the ar command that new files should be placed before *posname*.
c	Suppresses the message that is normally produced when *afile* is created.
i	Tells the ar command that new files should be placed before *posname*.
l	Places files in the local directory instead of the /tmp directory where they are normally placed.
o	Resets the last-modified date to the date recorded in the archive.
u	Replaces only those files with last-modified dates later than the archive files.
v	Gives a file-by-file description of the making of a new archive file from the old archive and the constituent files.

as – assembler

Syntax

as [-d124] [-L] [-W] [-V] [-J] [-R] [-t *directory*] [-o *objfile*] [*name...*]

Options

-d	Specifies number of bytes for offsets that involve forward or external references and have sizes unspecified in assembly language.
-J	Uses long branches to resolve jumps when byte-displacement branches are insufficient.
-L	Saves defined labels beginning with L, which are normally discarded.
-o	Specifies the name of the output file.
-R	Make initialized data segments read only, by concatenating them to the text segments.
-t	Specifies a directory other than the default /tmp to receive the temporary file.
-V	Uses virtual memory rather than a temporary file for immediate storage.
-W	Do not complain about errors.

at, batch – execute commands at a later time

Syntax

at *time* [*day*] [*file*]
at -r *job...*
at -l [*job*]
batch [*file*]

Options

-r	Removes jobs previously scheduled by at or batch.
-l	Used to obtain or verify the job numbers.

awk – pattern scanning and processing language

Syntax

awk [-Fc] {-f *prog*} {-} [*file...*]

Options

-	Used for standard input file.
-Fc	Sets interfield separator to named character.
-f*prog*	Uses prog file for patterns and actions.

basename – strip directory names from pathname

Syntax

basename *string* [*suffix*]

bc – interactive arithmetic language processor

Syntax

bc {-c} {-l} [*file...*]

Options

-c	Compiles input only.
-l	Names arbitrary precision math library.

bdiff – big file differential comparator

Syntax

bdiff *file1 file2* [*n*] [-s]

Options

-s	Suppresses normal diagnostic messages.

biff – be notified if mail arrives and who it is from

Syntax

biff [yn]

Options

-n	Disables notification that mail has arrived.

-y	Enables notification that mail has arrived.

binmail – send or receive mail among users

Syntax

/bin/mail [+] [-i] [*person...*]
/bin/mail [+] [-i] -f *file*

Options

-f	Displays mail messages contained in the specified file (next argument) in place of your mailbox file.
-i	Notifies mail to continue after interrupts.

cal – print calendar

Syntax

cal [*month*] *year*

calendar – calendar reminder service

Syntax

calendar [-]

Options

-	Functions for every user who has a calendar file in his login directory.

capsar – prepares a DOTS or DDIF document for transport in the Mail system

Syntax

capsar [-c] [-t] [-x{hTD}] [filename]

Options

-c	Create an encapsulated DOTS bodypart from filename.
-t	Write the message type of filename to standard output.
-xh	Extract mail header lines from filename.
-xT	Extract all text parts of filename to standard output.
-xD	Extract all DOTS bodyparts from filename.

cat – concatenate and print data

Syntax

cat [-*b*] [-*e*] [-*n*] [-*s*] [-*t*] [-*u*] [-*v*] *file...*

Options

-b	Ignores blank lines and precedes each output line with its line number.
-e	Displays a dollar sign ($) at the end of each output line.
-n	Precedes all outputlines (including blank lines) with line numbers.

-s	Squeezes adjacent blank lines from output and single spaces output.
-t	Displays non-printing characters (including tabs) in output.
-u	Unbuffers output.
-v	Displays non-printing characters (excluding tabs).

cb - C program beautifier

Syntax

cb

cc - C compiler

Syntax

cc [*option...*] *file...*

Options

-f	Specifies that computations involving only FFLOAT numbers be done in single precision and not promoted to double.
-g	Directs the compiler to produce additional symbol table information for dbx(1).
-O	Uses the object code optimizer.

cd - change current directory

Syntax

cd *directory*

cdoc - Compound document converter

Syntax

cdoc [-s *format*] [-d *format*] [-O *file*] [-o *outputfile*] *inputfile*

Options

-s	Specifies the format of the input (source) file.
-d	Specifies the format of the output (destination) file.
-O	Names n file that contains the processing options to be applied during conversion.
-o	Specifies the name of the output file.

cflow - generate C flow graph

Syntax

cflow [-r] [-ix] [-i_] [-d*num*] *files*

Options

-d	The *num* decimal integer indicates the depth at which the flow graph is cut off.
-i_	Includes names that begin with an underscore.
-ix	Includes external and static data symbols.
-r	Reverse the "caller:callee" relationship producing an inverted listing showing the callers of each function.

checknr – check nroff/troff files

Syntax

checknr [-s] [-f] [-a.x1.y1.x2.y2. ... .xn.yn] [-c.x1.x2.x3... .xn] [*file*...]

Options

-a	Allows additional pairs of macros to be added to the list.
-c	Defines commands otherwise complained about as undefined.
-f	Ignores \f font changes.
-s	Ignores \s font changes.

chfn – change system finger entry

Syntax

chfn [*loginname*]

chgrp – change file group

Syntax

chgrp [-f] *group file*...

Options

-f	Reports only system and usage messages.

chmod – change file mode

Syntax

chmod *mode file*...

chsh – change login shell

Syntax

chsh *name* [*shell*]

clear – clear terminal screen

Syntax

clear

cmp – compare file data

Syntax

cmp [-l] [-s] *file1 file2*

Options

-l	Long format: byte where difference occurs (decimal) and data differences (octal).
-s	Suppresses normal output and displays return code only.

col – filter reverse line feeds

Syntax

col [-*options*]

Options

- -b Assumes that the output device does not have backspacing.
- -f Suppresses moving half lines to the next full line.
- -h Suppresses conversion of white space to tabs.
- -p Forces through unchanged any unknown escape sequences that are found in its input.
- -x Suppresses conversion of white space to tabs (same as -h).

colcrt – filter nroff output for CRT previewing

yntax

colcrt [-] [-2] [*file...*]

Options

- - Suppresses all underlining.
- -2 Causes half-lines to be printed, double spacing the output.

colrm – remove columns from a file

Syntax

colrm [*startcol*[*endcol*]]

comm – compare sorted data

Syntax

comm [-[123]] *file1 file2*

Options

1 Suppresses column one: lines in file1 only.

2 Suppresses column two: lines in file2 only.

3 Suppresses column three: lines in file1 and file2.

compact, uncompact, ccat – compress and uncompress files, and cat them

Syntax

compact [*name...*]
uncompact [*name...*]
ccat [*file...*]

compress, uncompress, zcat – compress and expand data

Syntax

compress [-f] [-v] [-c] [-b *bits*] [*name* ...]
uncompress [-f] [-v] [-c] [*name* ...]
zcat [*name* ...]

Options

-f Forces compression of *name*.

-c Makes *compress/uncompress* write to standard output.

-b	Specifies the allowable *bits* limit.
-v	Displays the percent reduction of each file.

cp - copy file data

Syntax

cp [-i] [-r] *file1 file2*

cp [-i] [-r] *file... directory*

Options

-i	Prompts user with the name of file whenever the copy will cause an old file to be overwritten.
-r	Copies only to directories.

cpio - copy file archives in and out

Syntax

cpio -o [*keys*]

cpio -i [*keys*] [*patterns*]

cpio -p [*keys*] *directory*

Options

-i	Copies files that match the specified pattern.
-o	Copies out the specified files.
-p	Copies file into the specified directory.

cpp - the C language preprocessor

Syntax

/lib/cpp [*option* ...] [*ifile* [*ofile*]]

Options

-B	Strips C++-style comments (begin with // and end with newline).
-C	Passes along all comments, except those found on cpp directive lines.
-M	Generates dependency lists suitable for use with make(1) instead of the normal output.
-P	Preprocesses the input without producing the line control information used by the next pass of the C compiler.
-R	Permits recursion when a macro is expanded.
-U*name*	Removes any initial definition of *name* where *name* is a reserved symbol that is predefined by the preprocessor.
-D*name*	
-D*name*=*def*	Defines *name* as if by a #define directive.
-I*dir*	Changes the algorithm for searching for #include files whose names do not begin with a backslash (\ ) to look in *dir* before looking in the directories on the standard list.

csplit – context split

Syntax

csplit [**-s**] [**-k**] [**-f** *prefix*] *file arg1* [*...argn*]

Options

-s	Suppresses the printing of all character counts.
-k	Leaves previously created files intact.
-f*prefix*	Names the created files *prefix00...prefixn*.

ctags – create a tags file

Syntax

ctags [*options*] *name...*

Options

-a	Appends information to an existing tags file.
-B	Uses backward search patterns.
-F	Uses forward search patterns.
-t	Creates typedef tags.
-u	Updates the specified tags file.
-v	Generates an index listing function name, file name, and page number.
-w	Suppresses warning diagnostics and generates a listing.

ctod – combine DDIS objects into DOTS format

Syntax

ctod [**-x**] *object.ddis*

Options

-x	Specifies that ctod is to DOTS encode the input file without resolving any external references present in the file.

ctrace – C program debugger

Syntax

ctrace [*options*] [*file*]
ctc [*options*] [*file*]
cter [*options*] [*file*]

Options

-f*functions*	Trace only these *functions*.
-v*functions*	Trace all but these *functions*.

cut – cut out selected fields of each line of a file

Syntax

cut **-c***list* [*file1 file2...*]
cut **-f***list* [**-d***char*] [**-s**] [*file1 file2...*]

Options

list	Specifies ranges that must be a comma-separated list of integer field numbers in increasing order.
-c*list*	Specifies character positions to be cut out.
-f*list*	Speficies the fields to be cut out.
-d*char*	Uses the specified character as the field delimiter.
-s	Supresses lines with no delimiter characters.

cxref – generate C program cross reference

Syntax

cxref [*options*] *files*

Options

-c	Prints a combined cross-reference of all input files.
-D*name*	Defines *name* to processor, as if by *#define*.
-I*dir*	Searches named directory for #include files whose names do not begin with a backslash (\).
-l*x*	Abbreviation for library name /lib/libx.a where x is a string.
-o*file*	Directs output to named *file*.
-s	Operates silently; does not print input file names.

-t	Formats listing for 80-column width.
-U*name*	Removes any initial definition of *name*.
-w<*num*>	Width option which formats output no wider than columns (decimal).

date – print date and time

Syntax

date [[yy[mm[dd]]]hhmm[.ss][-[-]tttt][z]] [+format]

Options

n	Insert a new-line character
t	Insert a tab character
m	Month of year – 01 to 12
d	Day of month – 01 to 31
y	Last 2 digits of year – 00 to 99
D	Date as mm/dd/yy
H	Hour – 00 to 23
M	Minute – 00 to 59
S	Second – 00 to 59
T	Time as HH:MM:SS
u	Displays time in Greenwich Mean Time
j	Day of year – 001 to 366
w	Day of week – Sunday = 0
a	Abbreviated weekday – Sun to Sat
h	Abbreviated month – Jan to Dec
r	Time in AM/PM notation

dc – desktop calculator

Syntax

dc [*file*]

dd – copy and convert data

Syntax

dd [*option* = *value*...]

Options

if=*name*	Input file name.
of=*name*	Output file name.
ibs=*n*	Input block size, *n* bytes.
obs=*n*	Output block size, *n* bytes.
bs=*n*	Set both input and output block size to *n* bytes, superseding ibs and obs.
cbs=*n*	Conversion buffer size, *n* bytes.
skip=*n*	Skip *n* input records before starting to copy.
files=*n*	Copy *n* input files before terminating.
seek=*n*	Seek *n* records from beginning of output file before copying.
rbuf=*n*	Use *n* buffers for reading from those raw devices that support n-buffered I/O.
wbuf=*n*	Use *n* buffers for writing from those raw devices that support n-buffered I/O.

count=*n*	Copy only *n* input records.
conv=ascii	Convert EBCDIC to ASCII.
conv=ebcdic	Convert ASCII to EBCDIC.
conv=ibm	Slightly different map of ASCII to EBCDIC (see RESTRICTIONS).
conv=block	Convert variable length records to fixed length.
conv=unblock	Convert fixed length records to variable length.
conv=lcase	Map alphabetics to lower case.
conv=ucase	Map alphabetics to upper case.
conv=swab	Swap every pair of bytes.
conv=noerror	Do not stop processing on an error.
conv=sync	Pad every input record to *ibs*.
conv=nomulti	Disable multiple tape volumes.
conv=...	

deroff – remove formatting codes from text

Syntax

deroff [-w] *file*...

Options

-w	Generates word list (one word per line).

df – display free and used disk space

Syntax

df [-i] [-n] [*filesystem...*] [*file...*]

Options

-i	Also report the number of used and free inodes.
-n	Do not update the file system statistics stored in memory.

dgate - log in to a DECnet remote system through an intermediate ULTRIX DECnet host (gateway system)

Syntax

dgate *host*

diction,explain - print wordy sentences; thesaurus for diction

Syntax

diction [-ml] [-mm] [-n] [-f *pfile*] *file...*
explain

Options

-mm	Overrides default macro package -ms.
-ml	Causes deroff to skip lists.
-f*pfile*	Specifies pattern file in addition to default file or, default file can be suppressed with -n flag.

diff - differential file comparator

Syntax

diff [-l] [-r] [-s] [-cefh] [-b] *dir1 dir2*
diff [-cefh] [-b] *file1 file2*
diff [-D*string*] [-b] *file1 file2*

Options

-l	Displays output in long format.
-r	Recursively checks files in common subdirectories.
-s	Displays names of files that are the same.
-S*name*	Beginning with the specified file starts a directory in the middle.
-e	Writes output to an ed script.
-f	*Writes output in reverse order to a script.*
-c*n*	Displays specified number of conte. lines with each output line.
-h	Makes a hasty comparison.
-D*string*	Causes diff to create a merged version of *file1* and *file2* on the standard output.
-b	Ignores trailing blanks and other strings of blanks to compare equal.

diff3 - 3-way differential file comparison

Syntax

diff3 [-ex3] *file1 file2 file3*

Options

-3	Produces an ed editor script containing the changes between file1 and file2 that are to be incorporated into file3.
-e	Produces an ed editor script containing the changes between file2 and file3 that are to be incorporated into file1.
-x	Produces an ed editor script containing the changes among all three files.

diffmk – mark differences between files

Syntax

diffmk *name1 name2 name3*

dircmp – directory comparison

Syntax

dircmp [-d] [-s] [-w*n*] *dir...*

Options

-d	Compares the contents of files with the same name in both directories and output a list telling what must be changed in the two files to bring them into agreement.
-s	Suppresses messages about identical files.
-w*n*	Changes the width of the output line to *n* characters.

dirname – deliver directory names from pathname

Syntax

dirname *string*

domainname – display or set the name of the current domain for this system

Syntax

domainname [*nameofdomain*]

dtoc – unpack objects from a DOTS file

Syntax

dtoc [[*object.dots*] *directory*]

du – print amount of disk usage

Syntax

du [-s] [-a] [*name...*]

Options

-a	Displays the disk usage for each file.
-s	Displays a summary total only.

echo – echo arguments

Syntax

echo [-n] [*arg*...]

Options

-n	Suppresses newlines from output.

echo – echo arguments

Syntax

echo [arg] ...

error – analyze and disperse compiler error messages

Syntax

error [-n] [-s] [-q] [-v] [-t *suffixlist*] [-I *ignorefile*] [*name*]

Options

-I *ignorefile*	Ignore the functions listed in the specified file (next argument).
-n	Does not touch files and sends error messages to the standard output.

-q	Prompts before touching the source file.
-S	Shows error in unsorted order from the error file.
-s	Displays *statistics* for each error type.
-T	Terse output.
-t *suffixlist*	Does not touch those files that match the specified suffix.
-v	Invokes the vi editor on each file that had been touched.

expand, unexpand – expand tabs to spaces, and vice versa

Syntax

expand [-*tabstop*] [-*tabn*...] [*file*...]
unexpand [-a] [*file*...]

Options

-#	Sets tabstops the specified number of spaces (#) apart.
-a	When used with unexpand, compresses file by inserting tabs for two or more spaces.

expr – evaluate expressions

Syntax

expr *arg*...

extract – interactive string extract and replace

Syntax

extract [-i *ignorefile*] [-m *prefix*] [-n] [-p *patternfile*] [-s *string*] [-u] *filelist*

Options

-i	Specifies a new ignore file to be used to instruct extract to ignore specific text strings.
-m	Specifies a prefix to message numbers in the nl_ file and in the file.
-n	Create a new message source file for each input file.
-p	Specifies a new *pattern* file to be used.
-s	Specifies a string to be output at the start of the file.
-u	Use a message file produced by a previous run of strextract.

eyacc – modified yacc allowing much improved error recovery

Syntax

eyacc [-v] [*grammar*]

file – determine file type

Syntax

file [-c] [-f ffile] [-m mfile] *filename* ...

Options

-c	Checks the magic file for format errors by printing the internal representation of the magic file.
-f	Interprets the following argument to be a file containing the names of the files to be examined.
-m	Instructs *file* to use an alternate magic file.

find – find files

Syntax

find *pathname-list* *expression*

Options

-atime *n*	Tests true if the file has been accessed in *n* days.
-cpio device	Writes current file on *device* in cpio(5) format (5120-byte records).
-exec command	Tests true if specified command returns a 0 on exit.
-group gname	Tests true if group ID matches specified group name.
-inum *n*	Tests true if the file has inode number *n*.
-links *n*	Tests true if the file has *n* links.
-mount	Tests true if the current file is on the same file system as the current starting pathname.

-mtime *n*	Tests true if the file has been modified in *n* days.
-name filename	Tests true if the *filename* argument matches the current file name.
-newer file	Tests true if the current file has been modified more recently than the argument *file*.
-ok command	Executes specified command on standard output, then standard input is read and command executed only upon response y.
-perm onum	Tests true if file has specified octal number.
-print	Prints current pathname.
-size *n*	Tests true if the file is *n* blocks long (512 bytes per block).
-type *c*	Tests true if file is *c* type (*c* = b, block special file: c, character special file: d, directory: f, plain file: l, symbolic link: p, type port: s, type socket).
-user uname	Tests true if file owner is login name or numeric user ID.

finger – print user finger information

Syntax

finger [*options*] [*name...*]

Options

-b	Displays a briefer long form list of users.
-f	Disables printing of headers for short and quick outputs.
-h	Suppresses printing of the .project file.
-i	Displays list of users with idle times.
-l	Displays output in long format.
-m	Matches arguments only on user name.
-p	Suppresses printing of the .plan file.
-q	Displays list of users.
-s	Displays output in short format.
-w	Displays narrow short format of specified users.

fmt – simple text formatter

Syntax

fmt [*name...*]

fold – fold long lines for finite width output device

Syntax

fold [-*width*] [*file...*]

from – identifies sender of mail

Syntax

from [-f *mailbox*] [-s *sender*]

Options

-f *mailbox*	Uses specified file instead of your normal mail file.
-s *sender*	Prints mail headers for mail sent by specified sender.

fsplit – split a multi-routine Fortran file into individual files

Syntax

fsplit [-e *efile...*] [*file*]

Options

-e *efile*	Splits only specified subprogram units into separate files.

ftp – file transfer program

Syntax

ftp [-v] [-d] [-i] [-n] [-g] [host]

Options

-d	Enables debugging.
-g	Disables file name expansion.
-i	Disables interactive prompting during multiple file transfers.

-n	Disables autologin during an initial connection.
-v	Displays all responses from the remote server as well as all data transfer statistics.

gcore – get core images of running processes

Syntax

gcore *process-id...*

gencat – generate a formatted message catalog

Syntax

gencat [-h *hdrfile*] *catfile msgfile* ....

Options

-h	Causes gencat to generate a header file suitable for inclusion in the program source via a #include.

getopt – parse command options

Syntax

set -- `getopt optstring $*`

gprof – display call graph profile data

Syntax

gprof [*options*] [*a.out*[*gmon.out*...]]

Options

-a	Suppresses the printing of statically declared functions.
-b	Suppresses the printing of a description of each field in the profile.
-c	The static call graph of the program is discovered by a heuristic which examines the text space of the object file.
-E *name*	Suppresses the printing of the graph profile entry for routine *name* (and its descendants) as -e, above, and also excludes the time spent in *name* (and its descendants) from the total and percentage time computations.
-e *name*	Suppresses the printing of the graph profile entry for routine *name* and all its descendants More than one -e option may be given.
-F *name*	Prints the graph profile entry of only the routine *name* and its descendants (as -f, above) and also uses only the times of the printed routines in total time and percentage computations.
-f *name*	Prints the graph profile entry of only the specified routine *name* and its descendants.
-s	Produces a profile file *gmon.sum* is produced which represents the sum of the profile information in all the specified profile files.
-z	Displays routines which have zero usage (as indicated by call counts and accumulated time).

graph - draw a graph

Syntax

graph [*option*...]

Options

-a	Supplies abscissas automatically and uses next two arguments to set spacing and starting point.
-b	Breaks graph after each label in the input.
-c	Uses specified string (next argument) as label.
-g	Uses specified number (next argument) in setting up grid style: 0 (no grid), 1 (frame with ticks), and 2 (full grid).
-h	Uses specified number (next argument) as fraction of space for height.
-l	Uses specified string (next argument) as graph label.
-m	Uses specified number (next argument) in setting up line mode: 0 (disconnected) and 1 (connected).
-r	Uses specified number (next argument) as fraction of space to right before plotting.

-s	Saves screen (no erase) before plotting.
-t	Transposes vertical and horizontal axes.
-u	Uses specified number (next argument) as fraction of space to move up before plotting.
-w	Uses specified number (next argument) as fraction of space for width.
-x [l]	Determines x axis logarithmically.
-y [l]	Same as x but for y axis.

grep, egrep, fgrep – search file for regular expression

Syntax

grep [*option...*] *expression* [*file...*]
egrep [*option...*] [*expression*] [*file...*]
fgrep [*option...*] [*strings*] [*file*]

Options

-b	Precedes each output line with its block number.
-c	Produces count of matching lines only.
-e *expression*	Uses next argument as expression that begins with a minus (-).
-f *file*	Takes regular expression (egrep) or string list (fgrep) from *file*.
-i	Considers upper and lowercase letter identical in making comparisons (grep and fgrep only).

-l	Lists files with matching lines only once, separated by a new line.
-n	Precedes each matching line with its line number.
-s	Silent mode and nothing is printed (except error messages).
-v	Displays all lines that do not match specified expression.
-w	Searches for an expression as for a word (as if surrounded by '\<' and '\>').
-x	Prints exact lines matched in their entirety (fgrep only).

groups – show group memberships

Syntax

groups [*user*]

head – give first few lines

Syntax

head [*-count*] [*file...*]

hostid – set or print identifier of current host system

Syntax

hostid [*identifier*]

hostname – print system name

Syntax

hostname [*nameofhost*]

ic – compiler for language support database

Syntax

ic [**-D***name=def*] [**-U***name*] [**I***dir*] [**-v**] [**-o** *output*] [*source*]

Options

-D Defines name to the C preprocessor, as if #define name had been typed at the head of a source file.

-U Removes any initial preprocessor definition of name.

-I Searches for #include files in the named directory.

-o Names the ic output file output.

-v Gives statistics on the number of simple and double letters, the number of tables in the source and the size of the generated binary file.

id – print user and group IDs and names

Syntax

id

Indent – Indent and format C program source

Syntax

Indent *input* [*output*] [*flags*]

Options

-l*nnn*	Determines maximum length of output line.
-c*nnn*	Determines column in which comments start.
-cd*nnn*	Determines column in which comments on declarations start.
-i*nnn*	Determines number of spaces for one indentation level.
-dj,-ndj	Causes declarations to be left justified.
-v,-nv	**-v** turns on "verbose" mode, **-** turns it off.
-bc,-nbc	Forces newline after each comma in a declaration.
-d*nnn*	Controls the placement of comments which are not to the right of code.
-br,-bl	Specifying **-bl** causes complex statements to be lined up in a space order.

install – Install binaries

Syntax

install [**-c**] [**-m** *mode*] [**-o** *owner*] [**-g** *group*] [**-s**] *binary destination*

Options

-c	Moves or copies binary to *destination*.
-g *group*	Specifies a different group from group staff for *destination*.
-m *mode*	Specifies a different mode from the standard 755 for *destination*.
-o *owner*	Specifies a different owner from owner root for *destination*.
-s	Strips the binary after it is installed.

iostat - report I/O statistics

Syntax

iostat [*interval*[*count*]]

ipcrm - remove a message queue, semaphore set

Syntax

ipcrm [*options*]

Options

-q *msqid*	Removes the message queue identifier *msqid* from the system and destroys the message queue and data structure associated with it.
-m *shmid*	Removes the shared memory identifier *shmid* from the system.
-s *semid*	Removes the semaphore identifier *semid* from the system and destroys the set of semaphores and data structure associated with it.
-Q *msgkey*	Removes the message queue identifier, created with key *msgkey*, from the system and destroys the message queue and data structure associated with it.
-M *shmkey*	Removes the shared memory identifier, created with key *shmkey*, from the system.
-S *semkey*	Removes the semaphore identifier, created with key *semkey*, from the system and destroys the set of semaphores and data structure associated with it.

ipcs - report interprocess communication facilities status

Syntax

ipcs [*options*]

Options

-m	Displays information about active shared memory segments.
-q	Displays information about active message queues.
-s	Displays information about active semaphores.
-a	Uses all print *options* (shorthand notation for -b, -c, -o, -p and -t).
-b	Displays the biggest allowable size information.

-C	Uses the specified core file (next argument) in place of /dev/kmem.
-c	Displays creator's login name and group name.
-N	Uses the specified *namelist* (next argument) in place of /vmunix.
-o	Displays the outstanding usage information (number of messages in queue, size of each and number of processes attached to shared memory segments).
-p	Displays the process ID information.
-t	Displays all time statistics.
	The next 9 characters are interpreted as three sets of three bits each.

join - join files

Syntax

join [*options*] *file1 file2*

Options

-a*n*	Produces an additional line for unpaired lines from specified file *n*, where *n* is 1 or 2.
-e *s*	Uses specified replacement string for all empty output fields.
-j*n m*	Joins the *m*th field in the *n*th file.
-o *list*	Uses specified list for output line fields.
-t*c*	Sets tab character.

kill - send a signal to a process

Syntax

kill [-*sig*] *processid...*
kill -l

Options

-l	Lists signal names.

last - indicate last logins of users and teletypes

Syntax

last [-N] [*name...*] [*tty...*]

Options

-N	Limits the number of output lines to the specified number.

lastcomm - show last commands executed in reverse order

Syntax

lastcomm [*command name...*] [*user name...*] [*terminal name...*]

ld - link editor

Syntax

ld [*option...*] *file...*

Options

The -T option may be used as well, and is taken to mean that the newly linked segment commences at the corresponding address (which must be a multiple of 1024).

leave – remind you when you have to leave

Syntax

leave [hhmm]

lex – generate lexical analyzer

Syntax

lex [-tvfn] [*file*...]

Options

-f	Runs a faster compilation (does not pack resulting tables).
-n	Prints no summary information (default option).
-t	Writes to standard output instead of to file lex.yy.c.
-v	Prints one-line summary of generated statistics.

line – read one line

Syntax

line

lint – check C code

Syntax

lint [-abchnpuvxYz] *file*...

Options

-a	Report assignments of long values to int variables.
-b	Report break statements that cannot be reached.
-c	Complain about casts that have questionable portability.
-h	Apply a number of heuristic tests to attempt to find bugs, improve style, and reduce waste.
-n	Do not check compatibility against the standard library.
-p	Attempt to check portability to the IBM and GCOS dialects of C.
-u	Do not complain about functions and variables used and not defined, or defined and not used.
-v	Suppress complaints about unused arguments in functions.
-x	Report variables referred to be extern declarations, but never used.
-Y*environment*	Compiles C programs for *environment*.
-z	Do not complain about structures that are never defined (for example, using a structure pointer without knowing its contents.)

lk – link editor

Syntax

lk [*option...*] *file...*

Options

-D *number*	Sets data segment length.
-e *symbol*	Take the argument as the name of the entry point of the loaded program.
-H *number*	Takes number argument as a decimal integer, adds it to end of text, and starts data section at a higher address.
-K	Produces full load map, cross-referencing all defined symbols.
-L*dir*	Add *dir* to the list of directories in which libraries are searched for.
-l*x*	Abbreviation for the library name where *x* is a string.
-M	Produces full load map, consisting of a module and program section synopsis and symbol cross-reference.
-N	Do not make text portion read only or sharable.
-n	Arranges (by giving the output file a 0410 "magic number") that when the output file is executed, the text portion is read-only and shared among all users executing the file.
-o *name*	Takes the *name* argument after -o as the name of the lk output file, instead of a.out.
-S	Strips the output by removing all symbols except locals and globals.
-s	Removes the symbol table and relocation bits to save space.
-T *number*	Takes the argument as a hexadecimal number which sets the text segment origin.
-t	Displays the name of each file as it is processed.
-u *symbol*	Enters argument as undefined symbol in symbol table.
-X	Saves local symbols except for those whose names begin with 'L'.
-x	Suppresses saving nonglobal symbo' in output symbol table; enters only external symbols.
-Y*environment*	Adjust the magic number in the output file so that the program runs in the specified *environment* .
-y*sym*	Indicates each file in which *sym* appears, its type and whether the file defines or references it.
-z	Loads process on demand from the resulting executable file (413 format) rather than preloaded.

ln - link to a file

Syntax

ln [-f] [-s] *name1* [*name2*]
ln [-f] [-s] name ... directory

Options

-f	Suppresses all but the usage message.
-s	Creates a symbolic link.

lock - reserve a terminal

Syntax

lock

login - log in to a system

Syntax

login [*username*]

Options

-r	Used by the remote login server, rlogind(8c), to force login to enter into an initial connection protocol.
-P <*programname*>	Causes login to set it's standard input and output to be connected to the prompting program <*programname*>.
-C *string*	Allows the system to specify a command to be run using the user's shell.

logname - get login name

Syntax

logname

look - find lines in sorted data

Syntax

look [-df] *string* [*file*]

Options

-d	Uses dictionary order: only letters, digits, tabs and blanks can be compared.
-f	Folds uppercase to lowercase (compares equally).

indxbib, lookbib - build inverted index for a bibliography, lookup bibliographic references

Syntax

indxbib *database...*
lookbib *database*

lorder - determine relation for an object library

Syntax

lorder *file...*

lp – send requests to an LP line printer

Syntax

lp [-c] [-d *dest*] [-n *number*] [-] [*files*]

Options

-c	Makes copies of the *files* to be printed immediately when lp is invoked.
-d *dest*	Chooses *dest* as the printer that is to do the printing.
-n *number*	Prints *number* copies (default of 1) of the output.

lpq – spool queue examination program

Syntax

lpq [+ [n]] [-l] [-P*printer*] [*job #...*] [*user...*]

Options

+*n*	Displays spool queue.
-l	Displays information in long format.
-P	Displays information for the specified printer.

lpr – off line print

Syntax

lpr [*option...*] [*file...*]

Options

-p	Use pr(1) to format the files (equivalent to print).
-l	Use a filter which allows control characters to be printed and suppresses page breaks.
-t	The files are assumed to contain data from troff(1) (cat phototypesetter commands).
-n	The files are assumed to contain data from **ditroff** (device independent troff).
-d	The files are assumed to contain data from TeX (DVI format from Stanford).
-g	The files are assumed to contain standard plot data as produced by the plot(3x) routines (see also plot(1g)) for the filters used by the printer spooler).
-v	The files are assumed to contain a raster image for devices.
-c	The files are assumed to contain data produced by cifplot.
-f	Use a filter which interprets the first character of each line as a standard FORTRAN carriage control character.
-x	Transparent filter.
-r	Remove the file upon completion of spooling or upon completion of

	printing (with the -s option).
-m	Send mail upon completion.
-h	Suppress the printing of the burst page.
-P	Use to direct output to a specific printer
-C	The argument is the job classification for use on the burst page.
-J	The argument is the job name to print on the burst page.
-T	The argument is the title used by pr(1) instead of the file name.
-#*num*	**Produces multiple copies of output.**
-i	Causes the output to be indented the specified number of blank spaces.
-w	Takes the immediately following number to be the page width for pr.
-z	Takes the immediately following number to be the page length for pr.
-s	Uses symlink(2) to link data files rather than trying to copy them.
-1234	Specifies a font to be mounted on font position *i*.

lprm – remove jobs from line printer queue

Syntax

lprm [-P*printer*] [-] [*job #...*] [*user...*]

Options

-	Removes all jobs owned by you only.
- P *printer*	Removes jobs from specified printer.

ls – list and generate statistics for files

Syntax

ls [*options*] *name...*

Options

-1	Displays one entry per line.
-a	Displays all entries including those beginning with a period (.).
-C	Forces multicolumn output for pipe or filter.
-c	Sorts entries by time of modification.
-d	Displays names of directories only, not contents.
-F	Marks directories with trailing slash (/), sockets with a trailing equal sign (=), symbolic links with a trailing at sign (@), and executable files with a trailing asterisk (*).
-f	Displays names in the order they exist in directory.

-g Displays assigned group ID (used with -l only).

-i Displays the i-number for each file in the first column of the report.

-L Lists the information, if the file is a symbolic link, for the file or directory the link references rather than that for the link itself.

-l Lists the mode, number of links, owner, size in bytes, and time of last modification for each file.

The mode field consists of 11 characters.

d if the entry is a directory
b if the entry is a block-type special file
c if the entry is a character-type special file
l if the entry is a symbolic link
s if the entry is a socket
- if the entry is a plain file

The next 9 characters are interpreted as three sets of three characters each.

r if the file is readable
w if the file is writable
x if the file is executable
- if the indicated permission is not granted.

The group-execute permission character is given as s if the file has the set-group-id bit set; likewise, the user-execute permission character is given as s if the file has the set-user-id bit set.

The last character of the mode (normally 'x' or '-') is t if the 1000 bit of the mode is on.

-q Forces the printing of nongraphic characters in file names as the question mark character (?).

-R Recursively lists all subdirectories.

-r Sorts entries in reverse alphabetic or time order.

-s Displays the size in kilobytes of each file.

-t Sorts by time modified (latest first) instead of by name.

-u Uses the time of last access instead of last modification for sorting (with the -t option) or printing (with the -l option).

ltf - labeled tape facility

Syntax

ltf *option*[*keys*] *file*...

Options

-c Creates a new volume assigning an interchange file name to the files on the volume.

-H Displays help messages for all options and keys.

-t Lists each named file on the specified volume.

-x Extracts each named file from the volume to the user's current directory.

m4 - macro processor

Syntax

m4 [*files*]

make - maintain program groups

Syntax

make [**-f** *makefile*] [*option...*] [*name...*]

Options

-f	Uses specified file as input.
-i	Equals an .IGNORE: entry.
-k	Stops processing current entry on nonzero return, but continues with other branches that do not depend on that entry.
-n	Traces, prints, but does not update programs.
-r	Equals an initial .SUFFIXES: entry with no list.
-s	Equals a .SILENT: entry.
-t	Touches (updates) modification date of each target program only.

man - displays manual pages online

Syntax

man -k *keyword...*
man -f *page_title...*
man [**-**] [**-t**] [**-s**] [*1...8*] *page_title...*

Options

-k	Display one line summaries of each reference page that contains the specified keyword or keywords.
-f	Display one line summaries of each page title specified on the command line.
-	Squeeze multiple blank lines from output.
-s	Remove unnecessary blank lines.
-t	Phototypesets output using troff.

mdtar - multivolume archiver

Syntax

mdtar [*key*] [*name...*]

Options

-C	Changes directory to specified name.
c	Creates a new archive.
r	Writes the named files to the end of the archive.
t	If no file argument is given, all Generates archive table of contents.
u	Updates the current archive.
x	Extracts each specified file from the archive.
0...9	Selects unit number of the drive as an alternate disk drive.
A	Uses the specified number (next argument) as archive with which to begin the output.

b	Uses the specified number (next argument) as the blocking factor.
B	Forces output blocking to 20 blocks per record.
f	Uses the specified file (next argument) as the name of the archive.
F[F]	Operates in fast mode.
h	Saves a copy of the file (excludes symbolic links).
i	Ignores checksum errors found in the archive.
l	Displays an error message if all links to the files dumped cannot be resolved.
m	Does not restore file modification times.
o	Suppresses the normal directory information.
p	Restores the named files to their original modes, ignoring the present umask(2).
s	Uses specified number (next argument) as size of media in 512-byte blocks.
v	Displays detailed (verbose) information as it archives files.
w	Displays action to be taken for each file and prompts for confirmation.

mesg – allow or disallow messages

Syntax

mesg [n] [y]

mkdir – make a new directory

Syntax

mkdir *dirname...*

mkstr – create an error message file

Syntax

mkstr [-] *messagefile prefix file...*

Options

-	Places error messages at the end of specified message file.

more, page – display file data at your terminal

Syntax

more [-cdflsu] [-*n*] [+*linenumber*] [+/*pattern*] [*name...*]

page *more options*

Options

+*linenumber*	Start up at *linenumber*.
+/*pattern*	Start up two lines before the line containing the regular expression *pattern*.

-c	Begins each page at the top of the screen and erases each line just before it draws on it.
-d	Displays extended continuation prompt at end of each display.
-f	Counts logical text lines (does not fold long lines).
-l	Ignores line feeds (CTRL/Ls) and normally, pauses at line feeds.
-*n*	Specifies number of line more displays.
-s	Squeezes multiple blank lines from the output, producing only one blank line.
-u	Ignores all underlining in the data.
i <space>	Display *i* more lines, (or another screenful if no argument is given)
^D	Display 11 more lines (a "scroll").
d	Same as ^D (control-D)
*i*z	Same as typing a space except that *i*, if present, becomes the new window size.
*i*s	Skip *i* lines and print a screenful of lines
*i*f	Skip *i* screenfuls and print a screenful of lines
*i*b or *i*^B	Skip back *i* screenfuls and print a screenful of lines
q or Q	Exit from *more*.
=	Display the current line number.
v	Start up the editor vi at the current line.

h or ?	Help command; give a description of all the more commands.
/*expr*	Search for the *i*-th occurrence of the regular expression *expr*. If there are less than *i* occurrences of *expr*, and the input is a file (rather than a pipe), then the position in the file remains unchanged.
*i*n	Search for the *i*-th occurrence
'	(single quote) Go to the point from which the last search started.
!*command*	Invoke a shell with *command*.
i:n	skip to the *i*-th next file given in the command line (skips to last file if n doesn't make sense)
i:p	Skip to the *i*-th previous file given in the command line.
:f	Display the current file name and line number.
:q or :Q	Exit from more
.	(dot) Repeat the previous command.

mt - magnetic tape manipulating program

Syntax

mt [-f *tapename*] *command* [*count*]

Options

bsf	Backspace *count* files.
bsr	Backspace *count* records.
cache	Allows mt to use the cache buffer on a tape drive that has the cache buffer feature.

clhrdsf	Clear hardware/software problem.
clserex	Clear serious exception.
clsub	Clear subsystem.
eof, weof	Write *count* end-of-file marks at the current position on the tape.
eotdis	Disable end-of-tape detection.
eoten	Enable end-of-tape detection.
fsf	Forward-space *count* files.
fsr	Forward-space *count* records.
nocache	Disables the use of the cache buffer for any tape drive that has the cache buffer feature.
offline, rewoffl	Rewind the tape and place the tape unit off-line.
rewind	Rewind the tape.
status	Print status information about the tape unit.

mv – move or rename files

Syntax

mv [-i] [-f] [-] *file1 file2*

mv [-i] [-f] [-] *file... directory*

Options

-	Interprets all following arguments as file names to allow file names starting with a minus.
-f	Force.

-i	Interactive mode.

netstat – show network status

Syntax

netstat [-Aan] [-f *address_family*] [*system*] [*core*]

netstat [-himnrs] [-f *address_family*] [*system*] [*core*]

netstat [-n] [-I *interface*) *interval* [*system*] [*core*]

Options

-A	Displays the address of any associated protocol control blocks; used for debugging.
-a	Displays the information for all sockets.
-f *address_family*	Limits statistics or address control block reports to those of the specified *address family*.
-h	Displays the state of the IMP host table.
-I *interface*	Shows information only about this interface.
-i	Displays status information for autoconfigured interfaces.
-m	Displays information for the memory management routines The network manages a private share of memory.
-n	Displays network addresses as numbers.
-r	Displays the routing tables.

-s	Displays per-protocol statistics.
-t	Displays time until interface watchdog routine starts up (used only in conjunction with -i option).

newaliases – rebuild the data base for the mail aliases file

Syntax

newaliases

nice, nohup – execute a command at a lower priority

Syntax

nice [-*number*] *command* [*arguments*]

nohup *command* [*arguments*]

Options

-*number*	Increments the priority by a specified number up to a limit of 20.

nl – line numbering filter

Syntax

nl [-h *type*] [-b *type*] [-f *type*] [-v *start#*] [-i *incr*] [-p] [-l *num*] [-s *sep*] [-w *width*] [-n *format*] [-d *delim*] *file*

Options

-b *type*	Specifies which logical page body lines are to be numbered. The default *type* for logical page body is t (text lines numbered).
-h *type*	Same as -b *type* except for header.
-f *type*	Same as -b *type* except for footer.
-p	Do not restart numbering at logical page delimiters.
-v *start#*	The initial value used to number logical page lines.
-i *incr*	The increment value used to number logical page lines.
-s *sep*	The character used in separating the line number and the corresponding text line.
-w *width*	The number of characters used for the line number.
-n *format*	The line numbering format.
-l *num*	The number of blank lines to be considered as one.
-d *xx*	The delimiter characters specifying the start of a logical page section may be changed from the default characters (\:) to two user-specified characters.

nm – print program's name list

Syntax

nm [*options*] [*file...*]

Options

-a	Displays all symbols including debug symbol table.
-e	Prints only global (external) symbols.
-f	Displays all symbols including debug symbol table.
-g	Prints only global (external) symbols.
-n	Sorts numerically rather than alphabetically.
-o	Prepends file or archive element name to each output line.
-p	Prints symbolic table order and does not sort.
-r	Sorts in reverse order.
-u	Displays only undefined symbols.

nslookup - query BIND servers interactively

Syntax

/usr/ucb/nslookup [A] [*host* A [*server* A]]
nslookup - A [*server* A]

nsquery - name server query

Syntax

/usr/ucb/nsquery [lookup] [*host*] [*server*]

Options

lookup	Retrieves the host name, IP address, and aliases of the specified host.
host	Specifies the system for which you want host information.
server	Specifies the BIND server you want to query for the information.

od - create file octal dump

Syntax

od [*options*] [*file*] [*offset*] [*label*]

Options

-a	Interprets bytes as characters and display them with their ACSII names.
-b	Displays bytes as unsigned oct
-c	Displays bytes as ASCII characters.
-d	Displays short words as unsigned decimal.
-f	Displays long words as floating point.
-h	Displays short words as unsigned hexadecimal.
-i	Displays short words as signed decimal.
-l	Displays long words as signed decimal.
-o	Displays short words as unsigned octal.

-s[*n*]	Looks for strings of ASCII characters of *n* minimum length.
-v	Displays all data and indicates lines identical to the last line shown with an * in column 1.
-w[*n*]	Specifies the number of input bytes to be interpreted and displayed on each output line.
-x	Displays short words as hexadecimal.

pack, pcat, unpack – compress and expand files

Syntax

pack [-] [-f] *name...*

pcat *name...*

unpack *name...*

pagesize – print system page size

Syntax

pagesize

passwd – change your login password

Syntax

passwd [*name*]

paste – merge file data

Syntax

paste *file1 file2...*
paste -d*list file1 file2...*
paste -s [-d*list*] *file1 file2...*

Options

-	Used in place of any file name, to read a line from the standard input.
-d*list*	Replaces characters of all but last file with nontabs characters (default tab).
-s	Merges subsequent lines rather than one from each input file.

pc – Pascal compiler

Syntax

pc [*option*] *name...*

Options

-c	Suppresses loading and produce .o files from source files.
-g	Produces additional symbol table information for dbx(1).
-w	Suppresses warning messages.
-O	Invokes an object-code improver.
-o *output*	Names the final output file *output* instead of a.out.
-p	Prepares object files for profiling.

-S	Compiles the named program, and leave the assembler-language output on the corresponding file suffixed .s.
-C	Compiles code to perform runtime checks, verify assert calls, and initialize all variables to zero as in *pi*.
-b	Block buffers the file *output*.
-i*name*	Produces a listing for the specified procedures, functions and include files.
-l	Makes a program listing during translation.
-s	Accepts standard Pascal only and non-standard constructs cause warning diagnostics.
-t *directory*	Uses the given *directory* for compiler temporary files.
-z	Allows execution profiling with pxp by generating statement counters, and arranging for the creation of the profile data file pmon.out when the resulting object is executed.

pdx – pascal debugger

Syntax

pdx [-r] [*objfile*]

Options

-r	Causes obifile to be executed immediately.

pg – file perusal filter for soft-copy terminals

Syntax

pg [-p [-cefs] [files

Options

-number	Specifies the size (in lines) of the window that pg is to use instead of the default.
-p*string*	Uses the string as the prompt.
-c	Homes the cursor and clears the screen before displaying each page.
-e	Do not pause at the end of each file.
-s	Print all messages and prompt in standout mode (usually inverse video).
+*linenumber*	Starts at linenumber.
+/*pattern*/	Starts at the first line contain the regular expression, *pattern*.

pi – Pascal interpreter code translator

Syntax

pi [*options*] [-i *name...*] *name.p*

Options

-b	Block buffers the file *output*.
-i	Enables listing for specified procedures and functions and while processing specified include files.

-l	Creates a program listing while translating source.
-n	Begins each listed include file on a new page with a banner line.
-p	Suppresses control flow backtrace on error; suppresses statement limit counting.
-s	Accepts standard Pascal only; non-standard constructs cause warning diagnostics.
-t	Suppresses runtime tests of subrange variables and treat; treats assert statements as comments.
-u	Runs in card image mode; only the first 72 characters of input lines are used.
-w	Suppresses all warning diagnostics.
-z	Enables execution profiling with pxp by generating statement counters, and arranging for the creation of the profile data file *pmon.out* when the resulting object is executed.

pix – Pascal interpreter and executor

Syntax

pix [-blnpstuwz] [-i *name...*] *name.p* [*argument...*]

Options

-b	Block buffers the output.

-i*name*	Enables the listing for any specified procedures and functions, and while processing any specified include files.
-l	Creates a program listing while translating source.
-n	Begins each listed include file on a new page and with a banner line.
-p	Suppresses control flow backtraces on error.
-s	Accepts standard Pascal only.
-t	Suppresses runtime test of subrange variables.
-u	Runs in card image mode.
-w	Suppresses all warning diagnostics.
-z	Enables execution profiling.

plot – graphics filters

Syntax

plot [-T*terminal*[*raster*]] [-l#] [-w#] [-c#]

Options

-T*terminal*	Uses the specified terminal name as the terminal type for which plotting instructions are to be generated.
4020	Tektronix 4020 storage scope.
450	DASI Hyterm 450 terminal (diablo mechanism).
300	DASI 300 or GSI terminal (diablo mechanism).
300S	DASI 300S terminal (diablo mechanism).
ver	Versatec D1200A printer-plotter.

lvp16 DEC LVP16 Graphics Plotter.

hp7475a HP 7475A Graphics Plotter.

raster	Is a scan-converted temporary file that is sent directly to the plotter.
-l*#*	length of paper window in plotter units (unit scale)
-w*#*	width of paper window in plotter units (unit scale)
-c*#*	initial pen carousel to be used

pmerge - pascal file merger

Syntax

pmerge *name.p...*

pr - print files

Syntax

pr [*options*] [*files*]

Options

-a	Prints multi-column output across the page.
-b	Prints blank headers.
-d	Double-spaces the output.
-e *ck*	Expands *input* tabs to character positions k+1, 2*k+1, 3*k+1,...
-f	Uses form-feed character for new pages.
-h	Uses the next argument as the header to be printed instead of the file name.
-i *ck*	Replaces white space in *output* by inserting tabs to character positions k+1, 2*k+1, 3*k+1,...*n*k+1.
+k	Begins printing with page k (default is 1).
-k	Produces *k*-column output (default is 1).
-l *k*	Sets the length of a page to *k* lines.
-m	Merges and prints all files simultaneously, one per column (overrides the -k, and -a options).
-n *ck*	Numbers lines.
-o *k*	*Offsets each line by k character positions (default is 0).*
-p	Pauses before beginning each page the output is directed to a terminal.
-r	Suppresses diagnostic reports on failure to open files.
-s *c*	Separates columns by the single character *c* instead of by the appropriate number of spaces (default for *c* is a tab).
-t	Supresses the five-line identifying header and the five-line trailer normally supplied for each page.
-w *k*	Sets the width of a line to *k* character positions.

print – pr to the line printer

Syntax

print *file...*

printenv – display value of a shell variable

Syntax

printenv [*name*]

prmail – print out mail in the post office

Syntax

prmail [*user...*]

prof – profile an object file

Syntax

prof [-a] [-l] [-n] [-z] [-s] [-v] [*low*[-*high*]]] [*file1* [*file2...*]]

Options

-a	Displays all symbols rather than just external symbols.
-l	Displays output by symbol value.
-n	Displays output by number of calls.
-v	Produces graphic output for display by the plot(1g) filters.
-z	Routines having zero usage, as indicated by call counts and accumulated time, are printed in the output.

ps – print process status statistics

Syntax

ps [*options*] [*namelist*] [*core*]

Options

-#	Represents any given process number and must be the last option given.
-a	Displays information for processes executed from all user terminals.
-c	Displays command names which are stored internally in the system for accounting purposes rather than the command arguments, which are kept in the process address space.
-e	Displays the environment as well as the command arguments.
-g	Displays all processes.
-k	Uses file /vmcore in place of /dev/kmem and /dev/mem.
-l	Displays information in long format.
-s	Adds the size SSIZ of the kernel stack of each process to the basic output format for use by system maintainers.
-tx	Displays information for specified terminal only.
-u	Displays user-oriented output, which includes fields USER, %CPU, and

%MEM, SIZE.

-v	Displays process system time and user time in addition to cumulative time.
-w	Produces 132-column rather than 80 column output.
-x	Displays information for all processes, including those not executed from terminals.

ptx - create permuted index

Syntax

ptx [*option...*] [*input*[*output*]]

Options

-b *break*	Use the characters in the *break* file as separators.
-f	Folds upper and lower case letters for sorting.
-g *n*	Uses specified number as interfield gap.
-i *ignore*	Do not use as keywords any words given in the *ignore file*.
-o *only*	Use words listed only in the *only* file.
-r	Uses leading nonblanks as reference identifiers.
-t	Prepares the output for the phototypesetter.
-w *n*	Use the next argument, *n*, as the width of the output line.

pwd - print working directory

Syntax

pwd

px - Pascal code executor

Syntax

px [*obj*[*argument...*]]

pxp - Pascal execution profiler

Syntax

pxp [-acdefjnstuw_] [-23456789] [-z[*name...*]] *name.p*

Options

-_	Underscores all keywords.
-*d*	Uses the specified number (-*d*) the indentation unit.
-a	Displays all procedures (even those not executed).
-c	Uses the core file in generating the profiling data.
-d	Displays all declaration parts.
-e	Eliminates include directives when reformatting a file.
-f	Displays all parenthesized expression.
-j	Left justifies all procedures and functions.
-n	Begins a new page for each included file.

-s	Strips comments from the input text.
-t	Prints a table summarizing procedure and function call counts.
-u	Generates the output in card image format, using only the first 72 characters of input lines.
-w	Suppresses all warning diagnostics.
-z	Generate an execution profile for the specified modules (next arguments).

pxref – Pascal cross-reference program

Syntax

pxref [-] *name*

Options

-	Optional argument that suppresses the line numbered listing.

quota – display disk usage and limits

Syntax

quota [-qv] [*user*]

Options

-q	Prints a message that contains information only on file systems where usage is over quota.

-v	Displays users quotas on file systems where no storage is allocated.

ranlib – convert archives to random libraries

Syntax

ranlib *archive...*

rcp – remote file copy

Syntax

rcp [-p] *file1 file2*
rcp [-r] [-p] *file... directory*

Options

-p	Preserves the modification times and modes of the source files in its copies, ignoring the umask.
-r	Copies files in all subdirectories recursively, if the file to be copied is a directory.

refer – find and format bibliographic references

Syntax

refer [-a] [-b] [-c] [-e] [-f*n*] [-k*x*] [-l*m,n*] [-n] [-p *bib*] [-*skeys*] [-B*Lm*] [-P] [-S] [*file...*]

Options

-a*r*	Reverses order of first author names.
-B*Lm*	Bibliography mode.
-b	Creates bare entries: no flags, numbers, or labels.
-c*keys*	Capitalizes fields whose key letters are in string.
-e	Accumulates all references in one list.
-f*n*	Set the footnote number to *n* instead of the default of 1 (one).
-k*x*	Uses specified label in place of numbering for each reference data line beginning % *x* :.
-l*m,n*	Instead of numbering references, use labels made from the senior author's last name and the year of publication.
-p	Places punctuation marks .,:;?! after the reference signal, rather than before.
-n	Do not search the default file /usr/dict/papers/Ind.
-p*bib*	Specifies file to be searched before /usr/dict/papers.
-S	Produce references in the Natural or Social Science format.
-s*keys*	Uses specified key in sorting references.

reset – reset terminal mode

Syntax

reset

rev – reverse character positions in file data

Syntax

rev [*file...*]

rlogin – remote login

Syntax

rlogin *rhost* [-ec] [-8] [-L] [-l *username*]
rhost [-ec] [-8] [-L] [-l *username*]

Options

-8	Allows an 8-bit input data path at all times.
-ec	Uses the specified character as the rlogin escape character.
-l *username*	Logs you in as the specified user, not as your user login name.
-L	Runs session in litout mode.

rm, rmdir – remove (unlink) files or directories

Syntax

rm [-f] [-r] [-i] [-] *file-or-directory-name...*
rmdir *directory-name...*

Options

-	Specifies that the named files have names beginning with a minus (for example -myfile).
-f	Forces the removal of file or directory without first requesting confirmation.
-i	Prompts for yes or no response before removing each entry.
-r	Recursively removes all entries from the specified directory and, then, removes the entry for that directory from its parent directory.

rmail – route mail to users on remote systems

Syntax

rmail *user...*

roffbib – run off bibliographic database

Syntax

roffbib [*options*] [*file...*]

Options

-T*term*	Uses specified name as terminal type for which output is prepared.
-x	Suppresses the printing of abstracts.
-e	Formats text with equally spaced words, justified lines, and full resolution.
-h	Uses tabs in horizontal spacing to speed output and reduce output character count.
-n	Uses specified number (-n*N*) as first page to be printed.
-o	Uses specified list (-o*list*) as only pages to be printed.
-s	Stops after specified number of pages (-s*n*).
-m *mac*	Specifies a user-defined set of macros with space between -m and the macro file name.
-V	Sends output to the Versatec.
-Q	Queues output for the phototypesetter.
-r*aN*	Sets named register *a* to specified value *N*.

rsh – remote shell

Syntax

rsh host [-l *username*] [-n] *command*
host [-l *username*] [-n] *command*

Options

-l *username*	Logs you in as the specified user, not as your user login name.
-n	Redirects all command input to /dev/null.

ruptime – show host status of local machines

Syntax

ruptime [-a] [-l] [-t] [-u]

Options

-a	Users idle an hour or more are not counted unless this option is specified.
-d	Display only those hosts that are considered down.
-l	Sort the status list by load average.
-r	Show only hosts that are up and running.
-t	Sort the status list by uptime.
-u	Sort the status list by number of users.
-*nn*	Show only those hosts with *nn* or more users.

rwho – who is logged in on local machines

Syntax

rwho [-ah] [*users*]

Options

-a	Lists all users.
-h	Sorts users by host name.

s5make – maintain, update, and regenerate groups of programs

Syntax

s5make [-f *makefile*] [t[*names*]

Options

-b	Compatibility mode for old makefiles.
-d	Debug mode.
-e	Environment variables override assignments within makefiles.
-f *makefile*	Description file name determined by makefile as next argument.
-i	Ignore error codes returned by invoked commands.
-k	Abandons work on current entry, but continues on other branches that do not depend on that entry.
-m	Displays a memory map showing text, data, and stack.
-n	No execute mode.
-p	Displays the complete set of macro definitions and target descriptions.

-q	Question.
-r	Does not use built-in rules.
-s	Silent mode.
-t	Touches target files (causing them to be up-to-date) rather than issuing usual commands.

script – generate script of your terminal session

Syntax

script [-a] [*file*]

Options

-a Appends output to output file.

sed – stream text editor

Syntax

sed [-n] [-e *script*] [-f *sfile*] [*file*...]

Options

-e *script*
Uses specified file as input file of commands to be executed.

-f *sfile*
Uses specified file as input file of commands to be executed.

-n Suppresses all normal output.

size – print program's sizes

Syntax

size [*object*...]

sleep – suspend execution for a time

Syntax

sleep *time*

soelim – eliminate nroff source directives from nroff input

Syntax

soelim [*file*...]

Options

- File name corresponding to standard input.

sort – sort file data

Syntax

sort [*options*] [+*pos1* [-*pos2*]] [*file*...]

Options

-b Ignores leading blanks (spaces and tabs) in field comparisons.

-d Sorts data according to dictionary ordering: letters, digits, and blanks only.

-f Folds uppercase to lowercase while sorting.

-i Ignore characters outside the ASCII range 040-0176 in nonnumeric comparisons.

-n Sorts fields with numbers numerically.

-r Reverses the sense of comparisons.

-t*x* Uses specified character as field separator.

-c Checks sorting order and displays output only if out of order.

-m Merges previously sorted data.

-o*name*
Uses specified file as output file.

-T*dir*Uses specified directory to build temporary files.

-u Suppresses all duplicate entries.

sort5 – internationalized System 5 sort and/or merge files

Syntax

sort5 [*files*]

Options

-c	Checks that the input file is sorted according to the ordering rules; gives no output unless the file is out of order.
-m	Merges only; the input files are already sorted.
-u	Suppresses all but one in each set of lines having equal keys.
-o*output*	Specifies the name of an output file to use instead of the standard output.

-y*kmem*	Specifies the number of kilobytes of memory to use when sorting a file.
-z*recsz*	Records the size of the longest line read in the sort phase so buffers can be allocated during the merge phase.
	Sorts using tags.
-d	Specifies Dictionary order.
-f	Folds lower case letters into upper case.
-i	Ignores characters outside the ASCII range 040-0176 in nonnumeric comparisons.
-n	Sorts an initial numeric string, consisting of optional blanks, optional minus sign, and zero or more digits with optional decimal point, by arithmetic value
-r	Reverses the sense of comparisons.

sortbib – sort bibliographic database

Syntax

sortbib [-s*KEYS*] *database...*

Options

-s*KEYS*
Specifies new sort KEYS.

spell, spellin, spellout – check text for spelling errors

Syntax

spell [-v] [-b] [-x] [-d *hlist*] [+*local-file*] [-s *hstop*] [-h *spellhist*] [*file...*]

spellin [*list*]

spellout [-d] *list*

Options

-v Displays words not found in spelling list with all plausible derivations from spelling list.

-b Checks data according to British spelling.

-x Precedes each word with an equal sign (=) and displays all plausible derivations.

-d *hlist*
Specifies the file used for the spelling list.

-h *spellhist*
Specifies the file used as the history file.

-s *hstop*
Specifies the file used for the stop list.

+*local-file*
Removes words found in *local-file* from the output of the spell command.

spline – interpolate smooth curve

Syntax

spline [*option...*]

Options

-a Supplies abcissa automatically and uses specified number (next argument) for spacing.

-k Sets the boundary constant to the specified value (next argument).

-n Uses specified number (*n*) in calculating intervals between lower and upper limits.

-p Periodically produces output (matches derivates at ends).

-x Uses specified numbers (next arguments) as lower and upper limits.

split – split file into smaller files

Syntax

split [-*n*] [*file [name]*]

Options

- Uses standard input.

-n Writes specified number of lines to each output file.

strextract – batch string extraction

Syntax

strextract [-p *patternfile*] [-i *ignorefile*] [-d] [*file...]*

Options

- -i Specifies a new ignore file to be used to instruct strextract to ignore specific text strings.
- -p Specifies a new pattern file to be used.
- -d If this flag is set warnings are not printed for duplicate strings.

strings – print ASCII strings in program

Syntax

strings [-] [-o] [-*number*] *file...*

Options

- \- Looks through the entire object file for ASCII strings.

-*number*
Sets the minimum string length to specified number of characters and default is 4.

- -o Precedes each string with its file offset (octal).

strip – remove symbol table and relocation bits

Syntax

strip *name...*

strmerge – batch string replacement

Syntax

strmerge [-m *prefix*] [-p *patternfile*] [-s *string*] *file..*

Options

- -m Specifies a prefix to message numbers in the nl_ file and the file.
- -p Specifies a new *pattern* file to be used.
- -s Specifies a string to be output at the start of the file.

style – analyze surface characteristics of a document

Syntax

style [-ml] [-mm] [-a] [-e] [-l *num*] [-r *num*] [-p] [-P] *file...*

Options

- -a Displays all sentences with their length and readability index.
- -e Displays all sentences that begin with an expletive.

-l *num*
Displays all sentences longer than *num*.

- -ml Skips lists in document.
- -mm Overrides the default macro package -ms.
- -P Displays parts of speech of the words in the document.
- -p Displays all sentences that contain a passive verb.

-r *num*
Displays all sentences whose readability index is greater than *num*.

su – substitute a user ID

Syntax

su [*userid*]

Options

-f
Prevents csh(1) from executing the .cshrc file, making su start up faster.

- Simulates a full login.

sum – print object file's checksum

Syntax

sum *file*

symorder – rearrange name list

Syntax

symorder *orderlist symbolfile*

sync – update the super block

Syntax

/bin/sync

tabs – set tabs

Syntax

tabs [-n] [*terminal*]

Options

-n Does not indent left margin.

tail – print lines from file

Syntax

tail [±[number][lbc][fr]] [file]

talk – talk to another user

Syntax

talk *person* [*ttyname*]

tar – multivolume archiver

Syntax

tar [*key*] [*name...*]

Options

c Create a new archive on tape, disk, or file.

r Write the named files to the end of the archive.

t List the names of the files as they occur on the input archive.

u Add the named files to the archive if they are not there already or if they have been modified since they were last put in the archive.

x	Extract the named files from the archive.
0...9	Substitute a number for the device unit number, as in /dev/rmt#h.
A	Use next argument as archive number with which to begin output.
B	Force input and output blocking to 20 blocks/record.
D	Directory output in original tar style.
F[F]	Operate in fast mode.
H	Help mode.
M	Next arg specifies maximum archive number to be written and prints current archive number on output line.
N	No multi-archive, file splitting, or new header format on output.
O	Include file owner and group names in verbose output if present in archive header (t and x functions).
P	Used to specify POSIX format tapes.
S	Output User Group Standard archive format.
V	Display extended verbose information.
b	Use the next argument as the blocking factor for tape records.
d	Use /dev/rra1a as the default device (blocking factor of 10).
f	Use the next argument as the name of the archive instead of /def/rmt0h.
h	Save a copy of the actual file on the output device under the shmbolic link name, instead of placing the symbolic information on the output.
i	Ignore checksum errors found in the archive.
l	Complain if tar cannot resolve all of the links to the files dumped.
m	Do not restore the modification times.
o	Suppress the normal directory information.
p	Restore the named files to their original modes, ignoring the present umask(2).
s	Next argument specifies size of archive in 512-byte blocks.
v	Write the name of each file treated, preceded by the function letter, to diagnostic output.
w	Print the action to be taken, followed by file name, then wait for user confirmation.

tbl – format tables for nroff or troff

Syntax

tbl [*files*...]

tee – pipe output to terminal ar file

Syntax

tee [-i] [-a] [*file*...]

Options

- -a Appends input to existing files.
- -i Ignores interrupts.

telnet – user interface to the TELNET protocol

Syntax

telnet [*host*[*port*]]

test – test conditional expression

Syntax

test *expr*

test – condition evaluation command

Syntax

test expr
[expr]

tftp – trivial file transfer program

Syntax

tftp [*host*] [*port*]

time – time a command

Syntax

time *command*

tip, cu – connect to a remote system

Syntax

tip [-v] [-*speed*] *system-name*
tip [-v] [-*speed*] *phone-number*
cu *phone-number* [-t] [-s *speed*] [-a *acu*] [-l *line*] [-#]

Options

-*#* Uses specified speed (*#*) as baud rate.

-l Uses specified terminal line.

-v Displays all variable settings.

~CTRL/D
Drop the connection and exit (you may still be logged in on the remote machine).

~c [*name*]
Change directory to name (no argument causes a change to your home directory).

~! Escape to a shell (exiting the shell returns you to tip).

~> Copy file from local to remote.

~< Copy file from remote to local.

~p *from* [*to*]
Send a file to a remote UNIX host.

~t *Take a file from a remote UNIX host.*

~| Pipe the output from a remote command to a local UNIX process.

~# Send a BREAK to the remote system.

~s Sets a varinble.

~v Displays sets as they are made.

~CTRL/Z
Stop tip (only available with job control).

~? Displays a summary of the tilde escapes

The tip utility uses the file /etc/remote to find how to reach a particular system and to find out how it should operate while talking to the system.

touch – update access and modification times of a file

Syntax

touch [-amcf] [mmddhhmm[yy]] *files*

Options

-a	Causes touch to update the access time.
-c	Prevents touch from creating the file if it did not previously exist.
-f	Attempts to force the touch in spite of read and write restrictions on a file.
-m	Causes touch to update the modification time.

tr – translate characters

Syntax

tr [-cds] [*string1* [*string2*]]

Options

-c Translates complements: string1 to those not in string1.

-d Deletes all characters in string1 from output.

-s Squeezes succession of a character in string1 to one in output.

trace – trace system calls of programs

Syntax

trace [*options*] *cmd args...*

Options

-f *filename*
Puts dump in file *filename*.

-z Echos arguments only.

-c# Traces given PIDs and their children.

-g# Traces given groups only.

-p# Traces given PIDs only.

-s# Traces given system calls only.

-u# Traces given UIDs only.

trans – translation tool for use with message source files

Syntax

trans [-c] [-o *name*] *file.msf*

Options

-c Translate comment lines beginning with a dollar sign ($), including messages.

-o Call the output file *name*.

true, false – test for status

Syntax

true
false

tset – set terminal mode

Syntax

tset [*options*] [–m[*ident*] [*test baudrate*]:type] ... [type]
reset ...

Options

- Name of terminal is output on stndout, captured by the shell, and placed in the environment variable TERM.

–ec Uses the specified character as the erase character.

–I Suppresses transmitting terminal initialization strings.

–kc Uses the specified character as the kill character.

–n Initializes the "new" tty driver, if applicable.

–Q Suppresses erase and kill character message.

tsort – create topological sort

Syntax

tsort [*file*]

tty – print current terminal name

Syntax

tty [–s]

Options

–s Suppresses pathname.

ul – process underscores for terminal

Syntax

ul [–i] [–t *terminal*] [*name...*]

Options

–i Displays underscoring on separate line containing appropriate dashes (-).

–t *terminal*
Uses type of specified terminal in place your terminal's type.

uniq – report repeated lines in a file

Syntax

uniq [–udc[+n][–n]] [*input*[*output*]]

Options

–*n* Skips specified number of fields.

+*n* Skips specified number of characters in addition to fields.

–c Displays number of repetitions, if any, for each line.

- d Displays only lines that were repeated.
- u Displays only unique (nonrepeated) lines.

uptime – display system status

Syntax

uptime [-w]

users – print names of users who are logged in

Syntax

users

uucp, uulog, uuname – unix to unix copy

Syntax

uucp [*option...*] *source-file... destination-file*

uulog [*option...*]

uuname [*option...*]

Options

- d Creates all necessary directories for the file copy.
- c Uses the source file when copying out rather than copying the file to the spool directory.
- m Sends you mail when the copy is complete.
- n*rec* Sends mail to the recipient.
- W Expands only local files.
- s*sys* Displays information about work involving specified system.

- u*user* Displays information about work involving specified *user*.
- l Lists local system name.

uuencode, uudecode – encode/decode a binary file for transmission via mail

Syntax

uuencode [*file*] *remotedest* | mail *sys1*!*sys2*!..!*decode*
uudecode [*file*]

uusend – send a file to a remote host

Syntax

uusend [-m mode] sourcefile sys1!sys2!..!remotefile

Options

- m *mode* Specifies octal number for mode of file on the remote system.

uustat – uucp status inquiry and job control

Syntax

uustat [options]

Options

-*chour*
Removes entries older than specified hour.

-*jall* Reports status of all requests.

-k*jobn*
Kills specified job.

-m*mch*
Reports status of accessibility of machine *mch*.

-o*hour*
Reports status of requests which are older than specified hour.

-s*sys*Reports status of uucp requests for specified system.

-u*user*
Reports status of requests issued by specified user.

-v Invokes verbose printout option.

-y*hour*
Reports status of all requests that are younger than specified hour.

system
is the system in question

status_time
is the time the last status entry was made.

last_success_time
is the last time a connection was successfully made to this system.

statusis a self-explanatory description of the machine status.

uux – unix to unix command execution

Syntax

uux [-] *command-string*

Options

-c, Do not copy local file to the spool directory for transfer to the remote machine.

-g*grade*
Specifies the *grade* which is a single letter or number from 0 to 9, A to Z, or a to z.

-n Sends no notification to user.

-p, Reads stdin.

-r Queues the job, but does not start the file transfer.

-x*debug*
Produces debugging output on stdout.

-z Notify the user if the command fails.

vc – version control program

Syntax

vc [-a] [-t] [-c*char*] [-s] [*keyword=value*... *keyword=value*]

Options

-a Replaces the keywords surrounded by control characters in all text lines.

-c*char*
Specifies a control character to be used in place of :.

-s Suppresses all warning messages.

-t Ignores all characters from the beginning of the line to the first tab character.

vcc – VAX C compiler

Syntax

vcc [*option...*] *file...*

Options

The following is a list of the available options:

cross_reference	Generates a cross reference listing section
debug	Generates a loadable module for use with dbx
define	Assign a specified value to a name
g_float	Uses the G_floating point type
list	Generates a list file
machine_code	Generates the machine code listing section
object	Generates an object file with a specific name
optimize	Selects code optimization
show	Includes symbol and intermediate expansions
standard	Selects portability mode
undefine	Revokes the assignment of a value to a name
warnings	Disables warning or informational messages

-w Suppresses warning diagnostics.

- -DSYSTEM_FIVE is added to the vaxc command (or cpp command if -E is specified).
- The lk parameters -lc, -lcg, or -lc_p are preceded with -lcV, -lcVg, or -lcV_p (if not supressed by -b).
- The lk parameters -lm, -lmg, or -lmp are changed to -lmV, -lmVg, or -lmV_p (if present).

vdoc – Compound document viewer

Syntax

vdoc [-f *format*] [-O *file*] [-P *processing*] [-D *display*] *file*

Options

-f Specifies the format of the input file.

-O Names a file containing the processing options to be used by the front end converter.

-P Specifies the processing options to be applied during the veiwing of the file.

-D Specifies the content elements of the file to represented.

vmstat – report virtual memory statistics

Syntax

vmstat [-fsSzk] [*interval*[*count*]] [*namelist corefile*]

Options

-f Displays number of forks and vforks since system startup and number of pages of virtual memory involved in each kind of fork.

- -s Displays total number of paging-related events occurring since boot.
- -S Replaces the page reclaim (re) and pages attached (at) fields with processes swapped in (si) and processes swapped out (so).
- -k Allows a dump to be interrogated to print the contents of the sum structure (default).
- -z Zeroes out the sum structure.

w – display who is logged in and what they are doing

Syntax

w [-h] [-s] [-l] [*user*]

Options

- -d Outputs debug information.
- -h Suppresses the normal header from the output.
- -l Displays information in long format (default).
- -s Displays information in short format.
- -u Outputs the same information as the uptime command.

wait – wait for process completion

Syntax

wait [*pid*]

wall – write to all users

Syntax

wall

wc – count words, lines, and characters

Syntax

wc [-lwc] [*name...*]

Options

- -c Displays number of characters only.
- -l Displays number of lines only.
- -w Displays number of words only.

whatis – display command description

Syntax

whatis *command...*

whereis – locate source, binary, and or manual for program

Syntax

whereis [-sbm] [-u] [-SBM *dir...* -f] *name...*

Options

- -S dir
  Search for source files in specified directory.
- -B dir
  Search for binary files in given directory.

-M dir
Search for manual section files in given directory.

-b Searches only for binary files.

-f Terminates last directory list created from use of -S, -B or -M flags and signals the start of file names.

-m Searches only for manual section files.

-s Searches only for source files.

-u Searches for files that do not have one of binary, source or manual section files.

which - locate program file

Syntax

which [*name...*]

who - print who and where users are logged in

Syntax

who [*who-file*] [am i]

whoami - print your current login name

Syntax

whoami

write - write message to another user

Syntax

write *user* [*ttyname*]

xargs - construct argument list and execute command

Syntax

xargs [*flags*] [*command* [*initial-arguments*]]

Options

-l*number*	Execute the command for each non-empty number of lines of arguments from standard input.
-i*replstr*	Execute the command for each line from standard input, taking the entire line as a single argument and inserting it in *initial-arguments* for each occurance of *replstr*.
-n*number*	Execute the command using a many standard input arguments possible, up to the specified number.
-t	Echo the command and each constructed argument list to fd2 prior to their extraction.
-p	Asks the user whether or not the command should be executed each time it is invoked.
-x	Causes the command *xargs* to terminate if an argument list is greater than the specified size of characters.
-s*size*	The maximum size of each argument list is set to *size* characters.

-eeofstrR *The logical end-of-file string.*

xstr – extract strings from C program

Syntax

xstr [-c] [-] [*file*]

Options

- \- Reads stdin.
- -c Extracts strings from specified C source (next argument).

yacc – yet another compiler-compiler

Syntax

yacc [-vd] *grammar*

Options

- -d Writes all define statements to y.tab.h file.
- -v Writes description of parsing tables and report of grammatical conflicts to y.output file.

yes – be repetitively affirmative

Syntax

yes [arg]

ypcat – print values from a YP data base

Syntax

ypcat [-k] [-t] [-d *domainname*] *mname*
ypcat -x

Options

- -d *domainname* Displays information on the domain specified by *domainname*.
- -k Displays keys for maps in which values are null or key is not part of the value.
- -t Inhibits translation of *mname* to *mapname*.
- -x Displays map nickname table.

ypmatch - print the value of one or more keys from a yp map

Syntax

ypmatch [-d *domain*] [-k] [-t] *key...*\ *mname*
ypmatch -x

Options

- -d Displays key values for specified domain.
- -k Displays key, followed by a colon (:), before displaying value of the key.
- -t Inhibits translation of nickname to mapname.
- -x Displays map nickname table.

yppasswd – change login password in yellow pages map.

Syntax

yppasswd [*name*]

ypwhich – determine which host is the current YP server or map master.

Syntax

ypwhich [-d *domain*] [-V1] [-V2] [*hostname*]
ypwhich [-d *domain*] [-m *mname*] [-t]
ypwhich -x

Options

-V1 Identifies which server is serving v.1 YP protocol-speaking client processes.

-V2 Identifies which server is serving v.2 YP protocol-speaking client processes.

If neither version is specified, ypwhich attempts to locate the server that supplies the current v.2 services.

-d Uses *domain* instead of the current domain.

-m *mname*
Finds the master YP server for a map.

-t Inhibits nickname translation and is useful if there is a mapname identical to a nickname.

-x Displays the map nickname table.