Manual de UNIX
Rev 2.4

Jonathan Noel Tombs
Jorge Chávez Orzáez

Noviembre 1995

Introducción

En este manual hemos intentado recopilar nuestra experiencia en el sistema operativo UNIX. Está orientado fundamentalmente a usuarios poco experimentados (*lusers*[1]) en dicho sistema operativo, aunque se pretenden rellenar lagunas de conocimiento de otros usuarios "experimentados".

Somos conscientes de que todavía no está completo y puede contener gran número de imperfecciones así como errores ortográficos. No obstante no debe de confundirse el no entrar en detalle, con un error. Esto es, existen muchos comandos cuya explicación es demasiado simple, esto ha sido hecho a conciencia con objeto de no distraer al lector en detalles (que aunque para un lector avanzado puedan parecer fundamentales) los autores no consideran importantes para el objetivo planteado, no obstante se aceptan sugerencias.

Otro aspecto es la no traducción de gran parte de la terminología, también realizado con conciencia de ello, dado que la mayor parte de la documentación disponible se haya en lenguaje anglosajón.
Para cualquier comentario, *bug report*, sugerencia, etc.

`jon@www.esi.us.es`
`chavez@www.esi.us.es`

Agradecemos a todos los compañeros y alumnos que han colaborado en las correcciones y sugerencias de este manual, en particular a *Juan Toledo Cota* por sus múltiples (;-)) e interesantes correcciones.

Deseamos que este manual os sirva de ayuda para facilitar el acceso a este sistema operativo, y desmitifique el recelo que se sigue teniendo hacia él.

Copyright

[1] El diccionario **Jargon V 3.0** define:
:luser: /loo'zr/ n. A {user}; esp. one who is also a {loser}. ({luser} and {loser} are pronounced identically.) This word was coined around 1975 at MIT. Under ITS, when you first walked up to a terminal at MIT and typed Control-Z to get the computer's attention, it printed out some status information, including how many people were already using the computer; it might print "14 users", for example. Someone thought it would be a great joke to patch the system to print "14 losers" instead. There ensued a great controversy, as some of the users didn't particularly want to be called losers to their faces every time they used the computer. For a while several hackers struggled covertly, each changing the message behind the back of the others; any time you logged into the computer it was even money whether it would say "users" or "losers". Finally, someone tried the compromise "lusers", and it stuck. Later one of the ITS machines supported 'luser' as a request-for-help command. ITS died the death in mid-1990, except as a museum piece; the usage lives on, however, and the term 'luser' is often seen in program comments.

Indice

1 Introducción

UNIX es el sistema más usado en investigación científica, tiene una larga historia y muchas de sus ideas y método se encuentran en sistemas más modernos como DOS[2] y Windows.

Las características fundamentales del UNIX moderno son:

- Memoria Virtual:
  Memoria grande y lineal: Un programa en una máquina de 32 Bits puede acceder y usar direcciones de un rango de 4GB en un máquina de solo 4MB de RAM. El sistema sólo asigna memoria auténtica cuando le hace falta, en caso de falta de memoria de RAM, se utiliza el disco duro (*swap*).

- Multitarea (*Multitasking*):
  Cada programa con su propia "idea" de la memoria. Es imposible que un programa afecte a otro sin usar los servicios del sistema operativo. Si dos programas escriben en la misma dirección de memoria cada uno mantiene su propia idea de su contenido.

- Multiusuario:
  Más de una persona puede usar la máquina al mismo tiempo.
  Programas de otros usuarios continúan ejecutándose a pesar de que tú entres en la máquina.

- Casi todo tipo de dispositivo puede ser accedido como un fichero.

- Existen muchas utilidades diseñadas para que la salida de una pueda ser la entrada de la otra.

- Permite compartir dispositivos (como disco duro) entre una red de máquinas.

Por su naturaleza de multiusuario, **NUNCA** se debe apagar una máquina UNIX[3], ya que una máquina apagada sin razón puede matar trabajos de días, perder los últimos cambios de tus ficheros e ir degenerando dispositivos como el disco duro...
Entre los sistemas operativos UNIX actuales cabe destacar:

- **Linux**: disponible en la familia *x86*, las estaciones *Alpha* de Digital, la familia *68K*, estaciones *MIPS*, estaciones *SPARC* ...

- **SunOS**[4]: disponible para la familia *68K* así como para la familia *SPARC* de estaciones de trabajo SUN.

- **Solaris**[5]: disponible para la familia *SPARC* de SUN así como para la familia *x86*.

- **OSF1**[6]: disponible para *Alpha*

- **Ultrix**: disponible para *VAX* de Digital

[2] Pero para protegerse de los abogados de UNIX, introdujeron pequeños cambios para tener un interfaz distinto

[3] Incluyendo el caso en que la máquina es un PC normal corriendo Linux u otra versión de UNIX

[4] **SunOS 4.1.x** también se conoce como **Solaris 1**

[5] también conocido como **SunOS 5.x**, **Solaris 2** o *Slowaris* :-)

[6] también conocido como **Dec Unix**

- **SYSVR4**[7]: disponible para la familia *x86*, vax, ...
- **IRIX**: disponible para *MIPS*
- **AIX**[8]: disponible para *RS6000* de IBM y *PowerPC*.
- **WindowsNT**[9]: disponible para la familia *x86*, *Alpha* y *MIPS*.

2 Comenzando

En este apartado comentaremos las operaciones de comienzo y fin de una sesión así como la modificación de la contraseña (que a menudo no es la deseada por el usuario, y que por lo tanto puede olvidar con facilidad).

2.1 Terminales

Para iniciar una sesión es necesario poder acceder a un **terminal**. Pueden destacarse dos tipos de terminales:

- **terminal de texto**: Consta de una pantalla y de un teclado. Como indica su nombre, en la pantalla sólo es posible imprimir caracteres de texto.
- **terminal gráfico**: Consta de pantalla gráfica, teclado y ratón. Dicha pantalla suele ser de alta resolución y a menudo en color. Aunque al comenzar la sesión suelen estar en modo texto, una vez iniciada ésta, se puede trabajar en modo gráfico. En este modo se pueden emplear ventanas que emulan el comportamiento de un terminal de texto (`xterm`).

2.2 Login

El primer paso es encontrar un terminal libre donde aparezca el *login prompt* del sistema:

```
hostname login:
```

En este punto pueden ocurrir dos cosas:

- La pantalla está en blanco
  - comprobar que la pantalla esté encendida
  - pulsar la tecla `Return` para desactivar el protector de pantalla *screenblank*
- Otra persona ha dejado una sesión abierta. En este caso existe la posibilidad de intentar en otra máquina o bien finalizar la sesión de dicha persona (si ésta no se halla en las proximidades)

Una vez que se haya superado el paso anterior de encontrar el *login prompt* se procede con la introducción de tu **Username** al prompt de **login** y después tu contraseña (**password**) adecuado.

[7] También conocido como **Unixware** y **Novell-Unix**
[8] también conocido como **Aches** :-)
[9] Bill Gates dijo que "WindowsNT será el UNIX más usado en el mundo"

2.3 Passwords

El **password** puede ser cualquier secuencia de caracteres a tu elección. Deben seguirse las siguientes pautas:

- Debe ser fácil de recordar por uno mismo. Si se olvida, deberá pasarse un mal rato diciéndole al *System Administrator* que uno lo ha olvidado...
- Para evitar que alguna persona no deseada obtenga tu password y tenga libre acceso a los archivos de tu cuenta:
  - Las mayúsculas y minúsculas no son equivalentes sin embargo se recomienda que se cambie de una a otra
  - Los caracteres numéricos y no alfabéticos también ayudan. Debe tenerse sin embargo la precaución de usar caracteres alfanuméricos que se puedan encontrar en todos los terminales desde los que se pretenda acceder.
  - Las palabras de diccionario deben ser evitadas
- Debes cambiarlo si crees que tu password es conocido por otras personas, o descubres que algún intruso[10] está usando tu cuenta.
- El password debe de ser cambiado con *regularidad*.

La orden para cambiar el password en UNIX es `passwd`.
A menudo cuando existen varias máquinas que comparten recursos (disco duro, impresora, correo electrónico, ...), para facilitar la administración de dicho sistema se unifican los recursos de red (entre los que se hayan los usuarios de dicho sistema) en una base de datos común. Dicho sistema se conoce como **NIS** (*Network Information Service*)[11]. Si el sistema empleado dispone de este servicio, la modificación de la contraseña en una máquina supone la modificación en todas las máquinas que constituyan el **dominio NIS**.

2.4 Cerrando la sesión

Es importante que nunca se deje abierta una sesión, pues algún "gracioso" podría tener libre acceso a ficheros de tu propiedad y manipularlos de forma indeseable para ti.
Para evitar todo esto basta teclear `logout` ó `exit` y habrá acabado tu sesión de UNIX en dicha máquina[12]
Una vez que uno acabe su sesión, es conveniente que APAGUE EL MONITOR.

3 Almacenamiento de ficheros

Los sistemas de ficheros que son comunes a todos las máquinas son usualmente:

- /home – Espacio reservado para las cuentas de usuarios

[10] **intruso** es cualquier persona que no sea el usuario
[11] Antiguamente se conocía como **YP** (*Yellow Pages*), pero debido a un problema de marca registrada de United Kingdom of British Telecommunications se adoptaron las siglas **NIS**
[12] En caso de que se estuviera trabajando bajo *X-Windows* consultar la sección posterior del manual

- /bin, /usr/bin – Binarios (ejecutables) básicos de UNIX
- /usr/local – Zona con las aplicaciones no comunes a todos los sistemas UNIX, pero no por ello menos utilizadas...
  En dicha zona se pueden encontrar para algunas aplicaciones:
  - Información relacionada con dicha aplicación (en forma de páginas de manual, texto o bien ficheros Postscript)
  - Ficheros de ejemplo, *tutorials*, etc

4 Ordenes básicas

Para ejecutar un comando, basta con teclear su nombre (también debes tener permiso para hacerlo). Los argumentos empiezan normalmente con el carácter `-`

4.1 Ficheros y Directorios

En un sistema informático la información se encuentra en **ficheros** que contienen información (tabla de datos, texto ASCII, fuente en lenguaje C, ejecutable, imagen, figura, resultados de simulación, ...). Para organizar toda la información se dispone de una entidad denominada **directorio**, que permite el almacenamiento en su interior tanto de ficheros como de otros directorios[13]. Se dice que la estructura de directorios en UNIX es jerárquica o arborescente, debido a que todos los directorios nacen en un mismo punto (denominado directorio raíz). De hecho la zona donde uno trabaja es un nodo de esa estructura de directorios, pudiendo uno a su vez generar una estructura por debajo de ese punto.
Un fichero se encuentra situado siempre en un directorio y su acceso se realiza empleando el camino que conduce a él en el *Árbol de Directorios del Sistema*. Este camino es conocido como el *PATH*. El acceso a un fichero se puede realizar empleando:

- **Path Absoluto**, Aquel que empieza con `/`
  Por ejemplo : `/etc/printcap`
- **Path Relativo**, Aquel que **NO** empieza con `/`
  Por ejemplo : `examples/rc.cir`
- **Nombres de ficheros y directorios** pueden usar un máximo de 255 caracteres, cualquier combinación de letras y símbolos (el carácter `/` no se permite).

Los caracteres comodín pueden ser empleados para acceder a un conjunto de ficheros con características comunes. El signo ***** puede sustituir cualquier conjunto de caracteres[14] y el signo **?** cualquier carácter individual.
Por ejemplo[15]:

[13] Normalmente se acude a la imagen de una caja que puede contener informes o bien otros cajones, y así sucesivamente
[14] Incluido el punto '`.`', UNIX no es DOS
[15] `csh%` es el prompt en todos los ejemplos

```
csh% ls
f2c.1           flexdoc.1       rcmd.1          rptp.1          zforce.1
face_update.1   ftptool.1       rlab.1          rxvt.1          zip.1
faces.1         funzip.1        robot.1         zcat.1          zipinfo.1
flea.1          fvwm.1          rplay.1         zcmp.1          zmore.1
flex.1          rasttoppm.1     rplayd.1        zdiff.1         znew.1
csh% ls rp*
rplay.1         rplayd.1        rptp.1
csh% ls *e??
face_update.1   zforce.1        zmore.1
```

Los ficheros cuyo nombre comience por `.` se denominan **ocultos**, así por ejemplo en el directorio de partida de un usuario.

```
csh% ls -a ~user
.            .alias       .fvwmrc      .login       .xinitrc
..           .cshrc       .joverc      .profile
.Xdefaults   .enviroment  .kshrc       .tcshrc
```

Algunos caracteres especiales para el acceso a ficheros son:

.	Directorio actual
..	Directorio superior en el árbol
~	Directorio HOME[16]
~user	Directorio HOME del usuario *user*

4.2 Ordenes relacionadas con Directorios

$\boxed{\text{ls}}$

Este comando permite listar los ficheros de un determinado directorio. Si no se le suministra argumento, lista los ficheros y directorios en el directorio actual. Si se añade el nombre de un directorio el listado es del directorio suministrado.
Existen varias opciones que modifican su funcionamiento entre las que destacan:

- **-l** (Long listing) proporciona un listado extenso, que consta de los permisos[17] de cada fichero, el usuario el tamaño del fichero,...
- **-a** (list All) lista también los ficheros ocultos.
- **-r** (Recursive) lista recursivamente el contenido de todos los directorios que se encuentre.
- **-g** (list Group) lista el **grupo**[18] al que pertenece dicho fichero

[17] se comentará posteriormente este concepto

[18] cada usuario de Unix pertenece a uno o varios grupos, y cada fichero pertenece a un determinado usuario así como a un determinado grupo.

pwd

(*Print Working Directory*) Este comando proporciona el nombre del directorio actual

cd

(*Change Directory*) Permite moverse a través de la estructura de directorios. Si no se le proporciona argumento se provoca un salto al directorio `$HOME`.
El argumento puede ser un nombre absoluto o relativo de un directorio

mkdir

(*MaKe DIRectory*) Crea un directorio con el nombre (absoluto o relativo) proporcionado

rmdir

(*ReMove DIRectory*) Elimina un directorio con el nombre (absoluto o relativo) suministrado. Dicho directorio debe de estar vacío.

4.3 Visitando ficheros

Este conjunto de órdenes permite visualizar el contenido de un fichero sin modificar su contenido.

cat

muestra por pantalla el contenido de un fichero que se suministra como argumento.

more

esta orden es análoga a la anterior, pero permite la paginación.

less o **m**

es una versión mejorada del anterior. Aparte de que es más corto de teclear, permite una paginación correcta en ambas direcciones!!
Otra ventaja es que no lee el fichero entero antes de arrancar.

4.4 Copiando, moviendo y borrando ficheros

cp (*CoPy*)

copia un fichero/s con otro nombre y/o a otro directorio.
Veamos algunas opciones:

- **-i** (*interactive*), impide que la copia provoque una pérdida del fichero destino si éste existe[19].
- **-r** (*recursive*), copia un directorio y toda la estructura que cuelga de él.

mv (*MoVe*)

mover un fichero/s a otro nombre y/o a otro directorio.
Dispone de opciones análogas al caso anterior.

rm (*ReMove*)

borrar un fichero/s. En caso de que el argumento sea un directorio y se haya sumnistrado la opción **-r**, es posible borrar el directorio y todo su contenido.

4.5 Espacio de disco

Los usuarios disponen de una cuota de disco duro limitada, a continuación se comentan una serie de órdenes relacionadas con esta restricción:

[19] muchos sistemas tienen esta opción habilitada por omisión a través de un alias, para evitar equivocaciones de los Lusers

quota -v
muestra las cuotas de disco del usuario.
La opción **-v** permite ver las cuotas de un disco remoto.
du
(*Disk Usage*), permite ver el espacio de disco ocupado (en bloques de disco[20]) por el fichero o directorio suministrado como argumento. La opción **-s** impide que cuando se aplique recursividad en un directorio se muestren los subtotales
df
(*Disk Free*), muestra los sistemas de ficheros de los que dispone el sistema, con las cantidades totales/usadas/disponibles de cada uno
ln
Permite realizar un enlace (*link*) entre dos ficheros o directorios. Un enlace puede ser:

- *hard link*: se puede realizar sólo entre ficheros del mismo sistema de ficheros. El fichero enlazado apunta a la zona de disco donde se halla el fichero original. Por tanto, si se elimina el fichero original, el enlace sigue teniendo acceso a dicha información. Es el enlace por omisión.

- *symbolic link*: permite enlazar ficheros/directorios[21] de diferentes sistemas de ficheros. El fichero enlazado apunta al nombre del original. Así si se elimina el fichero original el enlace apunta hacia un nombre sin información asociada. Para realizar este tipo de enlace debe emplearse la opción **-s**.

Un enlace permite el uso de un fichero en otro directorio distinto del original sin necesidad de copiarlo, con el consiguiente ahorro de espacio.

4.6 Protección de ficheros

Dado que el sistema de ficheros UNIX es compartido por un conjunto de usuarios, surge el problema de la necesidad de privacidad. Sin embargo, dado que existen conjuntos de personas que trabajan en común, es necesario la posibilidad de que un conjunto de usuarios puedan tener acceso a una serie de ficheros (que puede estar limitado para el resto de usuarios).
Cada fichero y directorio del sistema dispone de un propietario, un grupo al que pertenece y unos **permisos**. Existen tres tipos fundamentales de permisos:

- **lectura** (**r**–*Read*): en el caso de un fichero significa poder examinar el contenido del mismo; en el caso de un directorio significa poder entrar en dicho directorio.

- **escritura** (**w**–*Write*): en el caso de un fichero significa poder modificar su contenido; en el caso de un directorio es crear un fichero o directorio en su interior.

- **ejecución** (**x**–*eXecute*): en el caso de un fichero significa que ese fichero se pueda ejecutar (binario o fichero de procedimientos); en el caso de un directorio es poder ejecutar alguna orden dentro de él.

Se distinguen tres grupos de personas sobre las que especificar permisos:

[20] 1 bloque normalmente es 1Kbyte
[21] debe hacerse notar que los directorios sólo pueden ser enlazados simbólicamente

- **user**: el usuario propietario del fichero
- **group**: el grupo propietario del fichero (excepto el usuario). Como ya se ha comentado, cada usuario puede pertenecer a uno o varios grupos y el fichero generado pertenece a uno de los mismos.
- **other**: el resto de los usuarios (excepto el usuario y los usuarios que pertenezcan al grupo)

También se puede emplear *all* que es la unión de todos los anteriores.
Para visualizar las protecciones de un fichero o directorio se emplea la orden `ls -l`, cuya salida es de la forma:

```
-rw-r--r--   ...otra información...    CD_list
```

Los 10 primeros caracteres muestran las protecciones de dicho fichero:

- El primer carácter indica el tipo de fichero de que se trata:
  - `-` fichero
  - `d` directorio
  - `l` enlace (*link*)
  - `c` dispositivo de caracteres (p.e. puerta serie)
  - `b` dispositivo de bloques (p.e. disco duro)
  - `s` socket (conexión de red)
  - `p` tubería (*pipe*)
- Los caracteres 2,3,4 son los permisos de usuario
- Los caracteres 5,6,7 son los permisos del grupo
- Los caracteres 8,9,10 son los permisos del resto de usuarios

Así en el ejemplo anterior `-rw-r--r--` se trata de un fichero donde el usuario puede leer y escribir, mientras que el grupo y el resto de usuarios sólo pueden leer. Estos suelen ser los permisos por omisión para un fichero creado por un usuario. Para un directorio los permisos por omisión suelen ser: `drwxr-xr-x` donde se permite al usuario "entrar" en el directorio y ejecutar órdenes desde él.

chmod

Esta orden permite modificar los permisos de un fichero.
`chmod` *permisos files*
Existen dos modos de especificar los permisos:

- Modo absoluto o modo numérico. Se realiza empleando un número que resulta de la OR de los siguientes modos:

400	lectura por el propietario.
200	escritura por el propietario.
100	ejecución (búsqueda) por el propietario.

040	lectura por el grupo.
020	escritura por el grupo.
010	ejecución (búsqueda) por el grupo.
004	lectura por el resto.
002	escritura por el resto.
001	ejecución (búsqueda) por el resto.
4000	*Set User ID*, cuando se executa este binario el proceso corre con los permisos del dueño del fichero.
2000	*Set Group ID* cuando se executa este binario el proceso corre en el mismo grupo que tiene el fichero.

Por ejemplo:
`chmod 640 *.txt`
Permite la lectura y escritura por el usuario, lectura para el grupo y ningún permiso para el resto, de un conjunto de ficheros que acaban en `.txt`

- Modo simbólico o literal. Se realiza empleando una cadena (o cadenas separadas por comas) para especificar los permisos. Esta cadena se compone de:
  `who operation permission`
  siendo:
  - `who` : es una combinación de:
    - **u** : user
    - **g** : group
    - **o** : others
    - **a** : all (equivalente a **ugo**)

    Si se omite este campo se supone **a**, con la restricción de no ir en contra de la máscara de creación (`umask`).
  - `operation`: es una de las siguientes operaciones:
    - + : añadir permiso
    - − : eliminar permiso
    - = : asignar permiso, el resto de permisos de la misma categoría se anulan.
  - `permission`: es una combinación de los caracteres:
    - **r** : read
    - **w** : write
    - **x** : execute
    - **X** : ejecución en un directorio o de un fichero que tuviera el permiso de ejecución en alguna de las clases de usuario.
    - **s** : en ejecución usar los permisos de dueño.
    - **t** : después de terminar ejecución, el programa continúa en memoria (cache).

Por ejemplo:
`chmod u+x tarea`
Permite la ejecución por parte del usuario[22] del fichero `tarea`.

[22] un error muy frecuente es la creación de un fichero de órdenes (*script file*) y olvidar permitir la ejecución del mismo.

`chmod u=rx,go=r *.txt`
permite la lectura y ejecución del usuario, y sólo la lectura por parte del grupo y el resto de usuarios.
La opción `-R` hace que la orden se efectúe recusivamente.

umask

Esta es una orden intrínseca del Shell que permite asignar los permisos que se desea tengan los ficheros y directorios por omisión.
El argumento que acompaña a la orden es un número octal que aplicará una XOR sobre los permisos por omisión (`rw-rw-rw-` para ficheros y `rwxrwxrwx` para directorios). El valor por omisión de la máscara es **077** que sólo habilita al usuario para lectura-escritura. Otro valor que se suele emplear es **022** que permite además al grupo y al resto la lectura.
Sin argumentos muestra el valor de la máscara.

chgrp

Cambia el grupo propietario de una serie de ficheros/directorios
`chgrp` *grupo files*
El usuario que efectúa esta orden debe de pertenecer al grupo mencionado.

id

Muestra la identificación del usuario[23], así como el conjunto de grupos a los que pertenece.

4.7 Filtros

Existe un conjunto de órdenes en UNIX que permiten el procesamiento de ficheros de texto. Se denominan **filtros** (*Unix Filters*) porque normalmente se trabaja empleando redirección recibiendo datos por su `stdin`[24] y retornándolos modificados por su `stdout`[25].

awk gawk

Es un procesador de ficheros de texto que permite la manipulación de las líneas de una forma procedural (i.e. con decisiones en función del contenido de la misma).
(Ejemplo)
Supongamos que tenemos un fichero `file` con dos columnas.
`awk " print $2, $1 " file`
Imprime esas dos columnas en orden inverso.

cat

Es el filtro más básico, copia la entrada a la salida.

cut

Para un fichero compuesto por columnas de datos, permite el borrado de un rango de columnas.

diff

Permite comparar el contenido de dos ficheros

find

Permite la búsqueda de un fichero en la estructura de directorios
`find .  -name file.dat -print`
Comenzando en el directorio actual recorre la estructura de directorios buscando el fichero `file.dat`, cuando lo encuentre imprime el path al mismo.
`find .  -name "*%" -exec rm '{}' \;`

[23] a pesar de que el usuario se identifica por una cadena denominada *username*, también existe un número denominado **UID** que es un identificativo numérico de dicho usuario
[24] entrada estándar
[25] salida estándar

Busca en la estructura de directorios un fichero que acabe en `%` y lo borra.
`xargs orden` repetir `orden` para cada argumento que se leer desde *stdin*. Permite uso muy eficiente de **find**.
`find . -name ''*.dat'' -print | xargs mv ../data`
Busca en la estructura de directorios todos los ficheros que acaben en `.dat`, y los mueve al directorio `../data`.

grep

Permite la búsqueda de una cadena en un fichero

head

Muestra las primeras líneas de un fichero.
`head -30 file` Muestra las 30 primeras líneas de `file`

tail

Muestra las últimas líneas de un fichero.
`tail -30 file` Muestra las 30 últimas líneas de `file`
`tail +30 file` Muestra desde la línea 30 en adelante de `file`

tar **gtar**

Este comando permite la creación/extracción de ficheros contenidos en un único fichero denominado `tarfile`. Este `tarfile` suele se una cinta magnetica, pero también puede ser un fichero.
Existen dos versiones: `tar` que es la versión que viene contenida generalmente en un SO Unix, y la versión GNU[26] del mismo `gtar`.
La acción a realizar viene controlada por el primer argumento:

- **c** (*Create*) creación
- **x** (*eXtract*) extracción
- **t** (*lisT*) mostrar contenido
- **r** añadir al final
- **u** añadir aquellos ficheros que no se hallen en el *tarfile* o que hayan sido modificados con posterioridad a la versión que aparece.

A continuación se colocan las opciones:

- **v** : Verbose
- **z** : comprimir/descomprimir el contenido (sólo disponible en `gtar`)
- **f** `device` : permite especificar un dispositivo para el tarfile (por omisión `/dev/rmt8`):
  - `-` el dispositivo es el `stdin/stdout`
  - **/dev/rst?** SCSI tape interface
  - **/dev/rmt?** half-inch magnetic tape interface
  - **/dev/fd?** floppy

[26]**GNU** es un acrónimo de: **'GNU's Not UNIX!'**. GNU es el nombre del producto de la *Free Software Foundation*, una organización dedicada a la creación de programas compatible con UNIX (y mejorado respecto a los estándars) y de libre distribución.

- **M** : Multivolumen (sólo disponible en `gtar`), es decir, usar varios diskettes (volúmenes) para almacenar un *tarfile*.
- **b** `N` : permite la especificación del tamaño de bloques `N` deseados. Un bloque es `Nx512` bytes y por omisión `N=20`.

Veamos algunos ejemplos:

`tar cvf simul.tar *.dat`	genera un fichero `simul.tar` que contiene todos los ficheros que terminen en `.dat` del directorio actual. A medida que se va realizando indica el tamaño en bloques de cada fichero añadido.
`gtar zcvf simul.tgz *.dat`	igual que en el caso anterior, pero el fichero generado `simul.tgz` ha sido comprimido empleando `gzip`.
`tar tvf simul.tar`	muestra los ficheros contenidos en el tarfile `simul.tar`
`tar xvf simul.tar`	extrae todos los ficheros contenidos en el tarfile `simul.tar`
`gtar cMbf 18k /dev/fd0 simulacion`	permite el almacenamiento en más de un diskette (i.e. volumen) de la información contenida en el directorio `simulacion`. La opción **b** y el argumento `18k` permite un mejor acceso al diskette, al especificarse el tamaño de bloques empleado por éste.

wc (*Word Count*)
Contabiliza el número de líneas, palabras y caracteres

4.8 Transferencia a diskettes.

La filosofía de diferentes unidades (`A:`, `B:`,...) difiere de la estructura única del sistema de ficheros que existe en Unix.
Son varias las alternativas que existen para la transferencia de información a diskette.

- Una posibilidad es disponer de una máquina DOS con **ftp** instalado y acceso a red. Empleando dicha utilidad se pueden intercambiar ficheros entre un sistema y el otro.
- Existe un conjunto de órdenes MTools disponible en multitud de sistemas, que permiten el acceso a diskettes en formato DOS de una forma muy eficiente.
  - **mcopy** `file file`
  - **mdir**

  - **mcd** `dirname`
  - **mformat**

  Para especificar el fichero que se encuentra en el diskette, el nombre del fichero se compone: `a:filename`. Si se desea emplear el caracter comodín para un conjunto de ficheros del diskette debe de rodearse de dobles comillas el mismo para evitar la actuación del Shell (p.e. `mcopy "a:*.dat" .`).
  La opción **-t** realiza la conversión necesaria entre UNIX y DOS, que se debe realizar SÓLO en ficheros de texto.

- IBM AIX ofrece las órdenes:
  - **doswrite** `file file`
  - **dosread** `file file`
  - **dosdir**
  - **dosformat**

  El nombre del fichero `file` que reside en el fichero puede estar en mayúsculas o minúsculas.
  La opción **-a** realiza la conversión necesaria entre UNIX y DOS, que se debe realizar SÓLO en ficheros de texto.

4.8.1 Unix y DOS

Cuando se transfieren ficheros de **texto** entre DOS y Unix sin las precauciones adecuadas pueden aparecer los siguientes problemas:

1. En DOS los nombres de los ficheros pueden tener un máximo de 8 caracteres y una extensión de 3 caracteres. En Unix no existe restricción respecto a la longitud del nombre, y aunque pueden llevar extensión, no es obligatorio.
2. El `Return` de DOS se compone de `Carriage Return` y `Line Feed`. Sin embargo en Unix sólo existe el **Carriage Return**.
   Así un fichero de Unix visto desde DOS parece una única línea. El caso inverso es la aparición del carácter `∧M` al final de cada línea.
3. La presencia de caracteres con código ASCII por encima del 127 (ASCII extendido) suele plantear problemas. Debido a que en DOS dicho código depende de la asignación hecha, que a su vez depende del país.

Para solucionar estos problemas se emplean las órdenes:

- `dos2unix dosfile unixfile`
  Que realiza la conversión de formato DOS a Unix
- `unix2dos unixfile dosfile`
  Que realiza la conversión de formato Unix a DOS

(NOTA) Estos comandos no admiten metacaracteres en su uso[27]. Para realizar la conversión de un conjunto de ficheros se podría emplear un comando interno del Shell, que en `csh` o `tcsh` sería:

```
foreach file (*.txt)
dos2unix $file $file.dos
end
```

4.9 Más Commandos

users **who** **w**
Ver quién está conectado en la máquina
rusers
Análogo al anterior pero para máquinas remotas (lanza un broadcast e imprime las contestaciones de las máquinas)
ping
ver si una máquina está conectada a red y si camino de Internet hasta la misma funciona correctamente.
rup
ver la carga de todas las máquinas de la red más "próxima".
finger
`finger user` muestra información[28] sobre el usuario `user` en la máquina local.
`finger user@hostname` muestra información sobre un usuario llamado `user` en una máquina `hostname`.
`finger @hostname` muestra los usuarios de una máquina.
cal
Muestra el calendario del mes actual
date
Muestra el día y la hora actual
leave
Alarma programable para que avise al llegar una determinada hora **strings**
Muestra las cadenas literales que tiene un fichero binario

5 Shells

UNIX soporta varios **intérpretes de comandos** o *Shells*, que ayudan a que tu interacción con el sistema sea lo más cómoda y amigable posible. La elección de cuál es el más cómodo es algo personal; en este punto sólo indicaremos los cinco más significativos:

- **sh** : Bourne SHell, el shell básico, no pensado para uso interactivo.
- **csh** : C-SHell, Shell con sintaxis como lenguaje "C".
  El fichero de configuración es `.cshrc` (en tu directorio `$HOME`).

[27] es decir no se puede ejecutar `dos2unix *.txt *.txt.dos`
[28] La información proporcionada es el nombre de completo del usuario (`GCOS`), las última sesión en dicha máquina, si ha leido o no su correo y el contenido de los ficheros `.plan` y `.project` del usuario.

- **tcsh** : alTernative C-Shell (Tenex-CSHell), con editor de línea de comandos.
  El fichero de configuración es `.tcshrc`, o en caso de no existir, `.cshrc` (en tu directorio `$HOME`
- **bash** : Bourne-Again Shell, con lo mejor de sh, ksh y tcsh.
  El fichero de configuración es `.bash_profile` si estás entrando en tu cuenta por primera vez (i.e. un login), y después el fichero `.bashrc` (en tu directorio `$HOME`).
- **ksh** : Korn SHell, sintaxis de sh con soporte para uso interactivo. Es el que establece por omisión AIX.
  El fichero de configuración es `.profile` en caso de login, en caso contrario el fichero con el nombre de la variable ENV, si existe.

Si queremos cambiar de shell en un momento dado, sólo será necesario que tecleemos el nombre del mismo y estaremos usando dicho shell. Si queremos usar de forma permanente otro shell del que tenemos asignado por omisión[29] podemos emplear la orden `chsh` que permite realizar esta acción.

En los ficheros de configuración se encuentran las definiciones de las variables de entorno (*enviroment variables*) como camino de búsqueda PATH, los "aliases" y otras configuraciones personales.
Veamos unos caracteres con especial significado para el Shell:

- `"` permite delimitar una cadena (constituyendo un único argumento) con espacios, permitiendo la expansión de $, ~, \, *, ‘
  Por ejemplo:
  `mcopy "a:*.txt" .` permite que la expansión del carácter comodín la realice el programa, y no el Shell[30]
- `‘`[31] ejecuta la orden delimitada y pone su resultado en el `stdout`
  Por ejemplo:
  `echo ‘pwd‘` imprime por pantalla el nombre del directorio actual.
- `’`[32] no expande $, ~, \, ‘ incluídos dentro de la cadena delimitada.
  Por ejemplo:
  `echo ’pwd’` imprime por pantalla la cadena `pwd`
- `(comando)` hace un *fork* (nuevo shell hijo para ejecutar un proceso) del comando delimitado.
- `;` permite la ejecución de más de una orden en una sóla línea de comando.

[29] Por omisión se suele asignar **tcsh**

[30] recuérdese que esta orden permite la copia de ficheros entre diskettes en formato DOS y un sistema Unix. Si el Shell realizara la expansión, pasaría como argumentos a la orden los ficheros del directorio actual que terminaran en `.txt`

[31] Esta tilde es la empleada en francés o inclinada hacia atrás.

[32] Esta tilde es la empleada en español o inclinada hacia delante.

5.1 Variables de Entorno

Las variables de entorno permiten la configuración por defecto de muchos programas (donde los programas buscan datos y tus preferencias) y se encuentran definidas en los ficheros de configuración anteriormente mencionados. Para referenciar a las variables poner el símbolo `$` delante, por ejemplo, para mostrar el camino de tu directorio por defecto:
`echo $HOME`
Las variables de entorno más importantes son:

- `HOME` – Tu directorio por defecto
- `PATH` – Tu camino de búsqueda, una lista de directorios separado con ':' para buscar programas
- `EDITOR` y/o `VISUAL` – Tu editor por defecto
- `DISPLAY` – Bajo el sistema de X windows, el nombre de máquina y pantalla que estás usando.
- `TERM` – Tu tipo de terminal[33].
- `SHELL` – Tu Shell por defecto
- `MANPATH` – Camino para buscar páginas de manuales
- `PAGER` – Programa de paginación de texto
- `TMPDIR` – Directorio para ficheros temporales

5.2 Redirección

Cuando el un programa espera que se teclee algo, aquello que el usuario teclea se conoce como el *Standard Input*: stdin. Los caracteres que el programa retorna por pantalla es lo que se conoce como *Standard Output*: stdout (o *Standard Error*: stderr[34]).
El signo `<` permite que un programa reciba el *stdin* desde un fichero en vez de la interacción con el usuario. Por ejemplo:

```
mail admin < file
```

Invoca el comando `mail` con argumento (destinatario del mail) `admin`, siendo el contenido del mensaje el contenido del fichero *file* en vez del texto que usualmente teclea el usuario.
Más a menudo aparece la necesidad de almacenar en un fichero la salida de un comando. Para ello se emplea el signo `>`.

```
man bash > file
```

Invoca el comando `man` con argumento (información deseada) `bash` pero indicando que la información debe ser almacenada en el fichero `file` en vez de ser mostrada por pantalla.

En otras ocasiones uno desea que la salida de un programa sea la entrada de otro. Esto se logra empleando los denominados *PIPES*, para ello se usa el signo `|`. Este signo permite que el *stdout* de un programa sea el *stdin* del siguiente.

```
zcat file.Z | more
```

[33] En la mayoría de los casos se trata de una emulación de `vt100`
[34] Si estos mensajes son de error

Invoca la orden de descompresión de `zcat`, y conducir dicho flujo de caracteres hacia el paginador `more`, de forma que podamos ver página a página el fichero descomprimido.

A parte de los símbolos mencionados existen otros que permiten acciones tales como:

- `>>` Añadir el *stdout* al final del fichero indicado (*append*)
- `>&` (`csh`, `tcsh` y `bash` sólo) Redireccionar el *stdout* y *stderr*.
- `>!` Igual que `>` pero con sobreescritura del fichero.
- `>>&` Igual que `>&` pero en modo *append*
- `>>!` Igual que `>>` pero con la adición que funciona también cuando el fichero no existente
- `2>` (`sh`, `ksh` y `bash` sólo)[35] Redireccionar el *stderr*.

5.3 CSH y TCSH

Son dos de los Shells interactivos más empleados. Una de las principales ventajas de `tcsh` es que permite la edición de la línea de comandos, y el acceso a la historia de órdenes usando las teclas de cursores.

5.3.1 Ejecución de comandos

- Si el comando introducido es propio del Shell (*built–in*), se ejecuta directamente.
- En caso contrario:
  - si el comando contiene `/` , el Shell lo considera un PATH e intenta resolverlo (entrar en cada directorio especificado para encontrar el comando).
  - en caso contrario el Shell busca en una tabla (*hash table*) que contiene los nombres de los comandos que se han encontrado en los directorios especificados en la variable `PATH`, cuando ha arrancado el Shell.

5.3.2 Aliases

Para facilitar la entrada de algunas órdenes o realizar operaciones complejas, los Shells interactivos permiten el uso de *aliases*. La orden `alias` permite ver que aliases hay definidos y también definir nuevos. Es corriente definir el alias `rm = 'rm -i'`, de esta forma la orden siempre pide confirmación para borrar un fichero. Si alguna vez quieres usar `rm` sin alias sólo hace falta poner delante el símbolo `\` , denominado *backslash* .
Por ejemplo `\rm` elimina los alias aplicados a `rm` .
Otro ejemplo bastante frecuente (en tcsh/csh) podría ser (debido a la complejidad de la orden):
`alias ffind 'find . -name \!* -print'`
Para emplearlo:
`ffind tema.txt` el resultado es la búsqueda recursiva a partir del directorio actual de un fichero que se llame `tema.txt`, mostrando el camino hasta el mismo.

[35] `csh` o `tcsh` no soportan redirección de *stderr*, y si quieres sólo redireccionar el *stderr* es necesario hacerlo de esta forma: `(comand > /dev/tty) >& fichero`

5.3.3 Comandos propios

Los comandos propios o intrínsecos (*Built-In Commands*) son aquellos que proporciona el propio Shell[36].

alias `name def`

asigna el nombre `name` al comando `def`.

foreach `var ( wordlist)`
commands
end

La variable `var` se asigna sucesivamente a los valores de cadena `wordlist`, y se ejecutan el conjunto de comandos. El contenido de dicha variable puede ser empleado en los comandos: `$var`.

history

muestra las últimas órdenes introducidas en el Shell
Algunos comandos relacionados con el *Command history* son:

- !! Repite la última orden
- !n Repite la orden `n`-ésima
- !string Repite la orden más reciente que empiece por la cadena `string`
- !?string Repite la orden más reciente que contenga la cadena `string`
- ∧str1∧str2 o !!:s/str1/str2/ (*substitute*) Repite la última orden reemplanzando la primera ocurrencia de la cadena `str1` por la cadena `str2`
- !!:gs/str1/str2/ (*global substitute*) Repite la última orden reemplazando todas las ocurrencias de la cadena `str1` por la cadena `str2`
- !$ Es el último argumento de la orden anterior que se haya tecleado.

pushd

Cambia de directorio, recordando el directorio actual.

popd

Retorna al directorio desde donde se hizo `pushd` la última vez.

repeat `count command`

Repite `count` veces el comando `command`.

rehash

Rehace la tabla de comandos (*hash table*)

set `variable = VALUE`

Asigna el valor de una variable del Shell.

set `variable`

Muestra el valor de la variable

setenv VARIABLE VALUE

Permite asignar el valor de una variable de entorno.

source `file`

[36] a diferencia de los comandos que provienen de un ejecutable situado en alguno de los directorios de la variable `PATH`

Ejecuta las órdenes del fichero `file` en el Shell actual.

unset `variable`

Desasigna el valor de una variable del Shell

unsetenv VARIABLE VALUE

Permite desasignar el valor de una variable de entorno.

umask `value`

Asigna la máscara para los permisos por omisión.

unalias `name`

Elimina un alias asignado.

5.3.4 Variables propias del Shell

Existe un conjunto de variables denominadas *shell variables*, que permiten modificar el funcionamiento del Shell.

`filec` (*FILE Completion*)

ES una variable *toggle* que permite que el Shell complete automáticamente el nombre de un fichero o un directorio[37]. Para ello, si el usuario introduce sólo unos cuantos caracteres de un fichero y pulsa el `TAB` el Shell completa dicho nombre. Si sólo existe una posibilidad, el completado es total y el Shell deja un espacio tras el nombre. En caso contrario hace sonar un pitido[38]. Pulsando `Ctrl-D` el Shell muestra las formas existentes para completar.

prompt

Es una variable de cadena que contiene el texto que aparece al principio de la línea de comandos.

savehist

permite definir el número de órdenes que se desea se almacenen al abandonar el shell. Esto permite recordar las órdenes que se ejecutaron en la sesión de otro dia.

5.4 SH y BASH

Sólo `bash` puede considerarse un Shell interactivo (`:-)`), permitiendo la edición de la línea de comandos, y el acceso a la historia de órdenes (*readline*). En uso normal (historia y editor de línea de comandos) BASH es compatible con TCSH y KSH.
El modo de completado (*file completion*) es automático (usando TAB sólo) si el SHELL es interactivo.

5.4.1 Comandos propios del Shell

`umask`, `source`, `pushd`, `popd`, `history`, `unalias`, `hash` [39] como en TCSH.

help Ayuda interna sobre los comandos del Shell.

`VARIABLE=VALUE`

Permite asignar el valor de una variable de entorno. Para que dicha variable sea "heredada"

---

[37] `tcsh` permite no sólo completar ficheros/directorios sino también comandos

[38] BEEP para los amigos

[39] En BASH/SH la *hash table* se va generando dinámicamente a medida que el usuario va empleando las órdenes. Así el arranque del shell es más rápido, y el uso de orden equivalente
`hash -r`
casi nunca hace falta

es necesario emplear:
export VARIABLE
o bien combinarlas:
export VARIABLE=VALUE
alias En BASH alias sólo sirve para substitución simple de una cadena por otra,
Por ejemplo:
`alias ls='ls -F'`.
Para crear aliases con argumentos se usan **funciones**. Las funciones se definen con () y los comandos a realizar entre llaves {}. El empleo de los argumentos se realiza mediante $0...N, siendo $# el número de argumentos.
Por ejemplo:

```
setenv() {
   if [ $# -gt 1 ]; then
      export \$1=``\$2''
   else
      env
   fi
}
```

Define una función igual que el `setenv` de TCSH. El siguiente defina un funciona equivalente al alias `ffind` de TCSH

```
ffind() {
    if [ $# != 1 ]; then
       echo Error, falta arguemnto
    else
       find . -name $1 -print
    fi
}
```

Las funciones pueden usar todas las órdenes de SH y UNIX y presenta una forma muy potente para construir aliases.

6 Ayuda y Documentación

Para obtener ayuda sobre comandos de UNIX, se puede emplear la ayuda *on-line*, en la forma de páginas de manual[40].
`man` *comando* : proporciona la ayuda sobre el *comando* deseado.
`man -k` *word* : proporciona las cabeceras de manuales donde aparezca la palabra *word*
por ejemplo, para leer el manual de los shells, puedes entrar:

`man sh csh tcsh bash`

la orden formatea las páginas y te permite leer los manuales en el orden pedido.

[40] En IBM/AIX la mejor orden es `info`

7 Procesos

En una máquina existen multitud de procesos que pueden estar ejecutándose simultánemente. La mayoría de ellos no corresponden a ninguna acción realizada por ti y no merecen que les prestes mayor atención.

Los programas suelen tener uno de estos dos modos de ejecución :

- **foreground**: Son aquellos procesos que requieren de la interacción y/o atención del usuario mientras se están ejecutando, o bien en una de sus fases de ejecución (i.e. Introducción de datos). Así por ejemplo una consulta de una página de manual es un proceso que debe ejecutarse claramente en *foreground*.
- **background**: Son aquellos procesos que no requieren de la interacción con el usuario para su ejecución. Si bien el usuario desearía estar informado cuando éste proceso termine. Un ejemplo de este caso sería la impresión de un fichero.

Sin embargo esta división que a primera vista pueda parecer tan clara y concisa, a menudo en la práctica aparece la necesidad de conmutar de un modo al otro, detención de tareas indeseadas, etc. Así por ejemplo puede darse el caso de que estemos leyendo una página de manual y de repente necesitemos ejecutar otra tarea.

Un proceso viene caracterizado por:

- *process number*
- *job number*

Veamos algunas de las órdenes más frecuentes para la manipulación de procesos:

- `comando &` Ejecución de un comando en el *background*[41]
- `Ctrl-Z` Detiene el proceso que estuviera ejecutándose en el *foreground* y lo coloca detenido en el *background*
- `Ctrl-C` Termina un proceso que estuviera ejecutándose en el *foreground*
- `Ctrl-\` Matar. Termina de forma definitiva un proceso que estuviera ejecutándose en el *foreground*
- `ps x` lista todos los procesos que pertenezcan al usuario, incluyendo los que no están asociados a un terminal
- `jobs` lista los procesos que se hayan ejecutado desde el shell actual, mostrando el *job number*
- `fg` *job number* pasa a ejecución en *foreground* un proceso que se hallase en *background*
- `bg` *job number* pasa a ejecución en *background* un proceso que se hallase detenido con `Ctrl-Z`

[41] Por omisión un comando se ejecuta siempre en el *foreground*.

- `kill` *process number* envía una señal[42] a un proceso UNIX.
  En particular `kill -KILL` envía la señal de término a un programa, pero no hace falta al ser la señal por defecto.

Cuando se intenta abandonar una sesión con algún proceso aún detenido en el background del Shell, se informa de ello con un mensaje del tipo:

`There are stopped jobs`

si no te importa, puedes intentar abandonarlo de nuevo y el Shell matará los jobs, o puedes utilizar `fg` o `bg` para terminar/dejarlos.

8 Editores

Un editor es un programa que permite crear y/o modificar un fichero. Existen multitud de editores diferentes, y al igual que ocurre con los shells, cada usuario tiene alguno de su predilección.
Mencionaremos algunos de los más conocidos:

- **vi** – El editor standard de UNIX.
- **emacs** – Editor muy configurable en Lisp.
  Existen multitud de **modos** para este editor (lector de mail, news, www,...) que lo convierten en un verdadero shell para multitud de usuarios. Las últimas versiones del mismo permiten la ejecución desde X-windows o terminal indistintamente con el mismo binario.
  Posee un tutorial en línea: `C-H t`
  El fichero de configuración personalizada es: `$HOME/.emacs`
- **jove** – Basado en Emacs, (Jonathan's[43] Own Version of Emacs).
  Posee tutorial en una utilidad asociada: `teachjove`
  El fichero de configuración personalizada es: `$HOME/.joverc`
- **jed** – Editor configurable en S-Lang.
  Permite la emulación de editores como **EMACS**, **EDT** [44] y **Wordstar**[45].
  Posee una ayuda en línea `C-H C-H`.
  El fichero de configuración personalizada es: `$HOME/.jedrc`
- **xedit** – Editor estándar del X Windows system.
- **textedit** – Standar Editor OpenLook del X Windows system.
- **xjed** – Versión de jed para el X Windows system.
  Presenta como ventaja que es capaz de funcionar en *modos* (lenguaje C, Fortran, TeX, Spice, Verilog, VHDL,...) reconociendo palabras clave y signos de puntuación, empleando un colorido distinto para ellos. El fichero de configuración personalizada es el de jed: `$HOME/.jedrc`

[42] Para ver las señales disponibles entra la orden `kill -l` (l==*list*).
[43] El Jonathan éste es otro :-)
[44] for VMS lusers
[45] Para los Turbo C lusers :-)

Dado que los editores **xedit** y **textedit** disponen de menús autoexplicativos, daremos a continuación unas ligeras nociones sobre el resto.

8.1 Editores modo `EMACS`

El editor GNU Emacs es uno de los que tienen mayor aceptación entre los usuarios de UNIX, estando disponible bajo licencia **GNU GPL**[46] para una gran cantidad de arquitecturas. Dentro de los "inconvenientes" que presenta es que NO viene por defecto incluído en la mayoría de los sistemas UNIX.

Este editor consta de tres zonas:

- La zona de edición: donde aparece el texto que está siendo editado y que ocupa la mayor parte de la pantalla.
- La zona de información: es una barra que esta situada en la penúltima línea de la pantalla.
- La zona de introducción de datos: es la última línea de la pantalla.

EMACS es un editor que permite la edición visual de un fichero (en constraste con el modo de edición de vi). La mayoría de los comandos de EMACS se realizan empleando la tecla de CONTROL o la tecla META[47]. Emplearemos la nomenclatura: `C-key` para indicar que la tecla `key` debe de ser pulsada junto con control y `M-key` para indicar que la tecla Meta debe de ser pulsada junto a `key` (en este último caso NO es necesario pulsar simultáneamente las teclas ESC y `key`, pudiendo pulsarse secuencialmente ESC y luego `key`).
A parte de las teclas rápidas que comentaremos, existen comandos que es posible ejecutar por nombre.

Ficheros

C-X C-F	cargar fichero	**C-X 2**	dividir ventana actual en 2 partes
C-X S	salvar fichero	**C-X 1**	sólo 1 ventana
C-X C-W	salvar con nombre	**C-X O**	conmutar siguiente ventana
C-X C-C	salir	**C-X B**	conmutar de buffer
C-X C-I	insertar fichero	**C-G**	aborta

(NOTA): Para salvar ficheros es válido tanto `C-X S` como `C-X C-S`, sin embargo debe evitarse esta última opción en los terminales de texto, pues `C-S` provoca la detención (*Stop Scroll*) del terminal. Para restaurar el terminal en este caso se emplea `C-Q`.

Comandos de movimiento

[46] La licencia de GNU, da el permiso de libre uso de los programas con su fuentes, pero los autores mantienen el *Copyright* (también conocido como *Copyleft*) y no es permitido distribuir los binarios sin acesso a sus fuentes, los programas derivados de dichos fuentes heredan la licencia GNU.

[47] Dado que la mayoría de los teclados actuales no poseen la tecla META se emplea como alternativa ESC

C-B	izquierda	**M-F**	avanza una palabra
C-F	derecha	**M-B**	retrocede una palabra
C-P	arriba una línea	**C-V**	avanza una página
C-N	abajo una ínea	**M-V**	retrocede una página
C-A	principio línea	**C-L**	refresca pantalla
C-E	fin de línea		
M->	fin documento		
M-<	principio documento		

Debe de comentarse que si el terminal lo acepta es posible el movimiento empleando las teclas de cursor

Comandos de inserción y borrado

Al ser un editor en modo visual, las modificaciones se pueden hacer en el texto sin necesidad de entrar en ningún modo especial.

Definición de regiones y reemplazo

C-space	Comienzo región	**C-S**	Búsqueda hasta fin texto
M-W	Copia región	**C-R**	Búsqueda hasta comienzo texto
C-W	Corta región	**M-Q**	Búsqueda y sustitución pide confirmación para sustituir (y/n)
C-Y	Pega región		
M-Y	Rotación regiones Aparecen las distintas regiones seleccionadas con anterioridad		

El editor conserva un conjunto de las últimas zonas seleccionadas durante la edición, pudiendo recuperarse una antigua a pesar de haber seleccionado una nueva zona (LIFO).

Definición de macros

C-X (	Comienza la definición de una macro
C-X)	Termina la definición de una macro
C-X E	Ejecuta una macro definida

Se entiende por macro a una sucesión de órdenes que se desea realizar.

Repetición

Cuando se desee repetir una órden un cierto número de veces se teclea previamente:

`ESC` *number*

Comandos

Aparte de los ya comentados existen muchas otras órdenes que no tienen necesariamente una tecla rápida (*bindkey*) asociada. Para su ejecución debe de teclearse previamente:

```
ESC X
```

y a continuación en la zona inferior de la pantalla se introduce el comando deseado. Empleando el TAB se puede completar dicho comando.

Es conveniente conocer las secuencias de control basico de `emacs` (`C-P, C-N, C-B, C-F, C-Y, C-W, C-K, C-T, C-D`) que también funcionan en el SHELL, muchos programas de texto y las ventanas de diálogo de las aplicaciones de X Windows. A su vez, los editores `jed, xjed, jove, xedit` también usan por defecto estas combinaciones.

9 El X windows system

El 'X windows system' es el sistema estándar de ventanas en estaciones de trabajo. Es corriente que el sistema de ventanas sea arrancado automáticamente cuando tu entras en tu cuenta. En caso contrario, la orden para arrancarlo es `startx`.
En el sistema X-windows deben distinguirse dos conceptos:

- *server* : Es un programa que se encarga de escribir en el dispositivo de vídeo y de capturar las entradas (por teclado, ratón, etc). Asimismo se encarga de mantener los recursos y preferencias de las aplicaciones.
  Sólo puede existir un server para cada pantalla!!
- *client* : Es cualquier aplicación que se ejecute en el sistema X Windows.
  No hay límite (en principio) en el número de clientes que pueden estarse ejecutando simultáneamente. Los clientes pueden ser locales o remotos.
  Window Manager (WM) Es un cliente con "privilegios especiales": Controla el comportamiento (forma,tamaño,..) del resto de clientes.
  Existen varios, destacando :
  - **fvwm** : *F* Virtual Window Manager*, el instalado por omisión.
  - **olwm** : *Open Look Window Manager*, propio de *SUN*
  - **twm** : *Tab Window Manager*, suministrado con la distribución **X11R*** del *MIT*
  - **mwm** : *Motif Window Manager*, suministrado con el **OSF/Motif** Toolkit

  El 'look and feel' (o GUI) de X windows es muy configurable, y puede parecer muy distinto, pero esto se debe al WM que se esté usando, no que las aplicaciones sean distintas.

Para configurar tu sesión es necesario saber qué programas estas usando y ver las páginas de manual. Los ficheros pricipales son:

`.xinitrc` ó `.xsession` fichero leído al arrancar X windows. Aqui se pueden definir los programas que aparecen al inicio de tu sesión.

`.fvwmrc` fichero de configuración del `fvwm`. Ver las páginas de `fvwm`.

`.olwmrc` fichero de configuración del `olwm`. Ver las páginas del manual de `olwm`.

`.Xdefaults` Configuración general de las aplicaciones de X windows. Aquí puedes definir los *Resources* que encontrás en los manuales de las aplicaciones de X.

`.rhosts` Fichero no de X windows, pero de permiso de acceso remoto, contiene los nombres de otras máquinas desde donde puedes *login* sin uso de un password. Ver `man rhosts`.

En caso de que tengas que correr una aplicación de X que no esté disponible en la máquina que estas usando, eso no representa un problema. Las ordenes necesarias son (por ejemplo para arrancar un `xterm` remoto):

```
unix1% xhost +unix2              #permite ventanas desde maquina unix2
unix1% rlogin unix2              #login remoto a otra maquina
Password:                        #Dar tu password (si es necesario)
unix2% setenv DISPLAY unix1:0    #definir el camino a tu pantalla
unix2% xterm                     #arranca la aplicacion en tu pantalla
```

Si todo está configurado correctamente, es posible que no haga falta dar un password (ver el fichero `.rhosts`), y en este caso puedes utilizar la orden `rcmd` para hacelo. Por ejemplo, lo de arriba se haría con:

`rcmd unix2 xterm`

Cuando quieres salir, normalmente puedes encontrar la opción SALIR, en un menú en la zona libre de la pantalla. Es corriente ver errores de tipo `'IO error'` al salir. Esto es porque has terminado tu sesión antes de terminar todos los programas que tienes corriendo. Ignora este tipo de error.

9.1 Uso del ratón

El ratón es un dispositivo esencial en el uso de programas X, sin embargo la función que realiza en cada uno de ellos no está normalizada...
Comentaremos la pauta seguida por la mayoría de las aplicaciones, pero debe tenerse presente que es muy frecuente encontrar aplicaciones que no las respetan[48].

- **Botón izquierdo** (LB): Seleccionar. Comienza el bloque de selección.
- **Botón central** (MB): Pegar. Copia la selección en la posición del cursor.
- **Botón derecho** (RB): Ajustar. Delimita la selección.

Existen dos modos para determinar cuál es la **ventana activa** (aquella que recibe las entradas de teclado):

- *Focus Follows Mouse*: La ventana que contenga al ratón es la que es activa. Es el modo por omisión
- *Click To Focus*[49]: La ventana seleccionada es la activa.

El modo que esté activo depende de la configuración del *Window Manager*.

[48] Las aplicaciones que son conscientes de un uso *anormal* y están relizadas por programadores inteligentes, muestran en pantalla la función de cada botón cuando son posibles varias alternativas

[49] Es el modo empleado en Microsoft Windows

9.2 Algunas Aplicaciones X

Pasemos a continuación a enumerar algunas de las aplicaciones X de uso más común:

- `xterm`: Es un emulador de terminal. Permite seleccionar el tamaño de la letra empleada (Ctrl-RB), la adición de una barra de *scroll* (Ctr-MB), etc.
- `xclock`: Reloj
- `xcalc`: Calculadora
- `xedit`: Editor de texto
- `xman`: Páginas de Manual
- `xfig`: Editor de gráficos vectoriales
- `xvgr`: Manipulador de datos (gráficas, procesamiento, etc.)
- `xspread`: Hoja de Cálculo
- `xdbx`: Debugger visual
- `xv`: Visualizador de imágenes en diferentes formatos gráficos (GIF, TIFF, PPM, ...)

10 Internet

En esta sección denominaremos `unix1` a la máquina local (desde donde ejecutamos la orden) y `unix2` a la máquina remota (con la que interaccionamos). Ambos son los *hostnames* de las respectivas máquinas.
Existen algunos conceptos que previamente debemos comentar:

- **IP-number**: es un conjunto de 4 números separados por puntos (p.e. `150.214.140.16`) que se asocia a cada máquina. No puede haber dos máquinas conectadas en la misma red con el mismo número.
- **hostname** Es el nombre que tiene asociada la máquina (p.e. `bart`). A este nombre se le suelen añadir una serie de sufijos separados por puntos que constituye el denominado **dominio** (p.e. `bart.esi.us.es`). Una máquina por tanto puede tener más de un nombre reconocido (se habla en este caso de **alias**). Se denomina **resolution** a la identificación entre un *hostname* y el *IP-number* correspondiente. La consulta se realiza inicialmente en el fichero `/etc/hosts`, donde normalmente se guardan las identificaciones de las máquinas más comunmente empleadas. En caso de que no se lograse se accede al servicio **DNS** (*Domain Name Service*), que permite la identificación (*resolution*) entre un *hostname* y un *IP-number*.
  Deben hacerse un par de observaciones:
  - En el caso de que un conjunto de máquinas estén empleando NIS (también conocido como *yellow pages*), el acceso al fichero `/etc/hosts` sólo se realiza en el server de este servicio cuando se rehacen las páginas de NIS.

- En el caso de algunos sistemas operativos[50] el acceso al servicio DNS (si se habilita) se antepone al acceso al fichero `/etc/hosts`

- **mail-address** Es el nombre que se emplea para enviar correo electrónico. Este nombre puede coincidir con el nombre de una máquina, pero se suele definir como un alias (con objeto de que la dirección no deba de cambiarse si la máquina se estropea).

10.1 Acceso a la red

Existen muchos programas para la conexión de la red, los más usados son:

- `rlogin -l nombre unix2`
  (Remote login), hace un login a la máquina `unix2` como el usuario `nombre` (por defecto, sin los argumentos `-l nombre rlogin` usa el nombre de tu cuenta local). Normalmente `rlogin` pide el password de la cuenta remota, pero con el uso del fichero `.rhosts` o `/etc/hosts.equiv` esto no es siempre necesario.

- `rsh -l nombre unix2 orden ...`
  (remote shell), ejecuta la orden `orden` en la máquina `unix2` como usuario `nombre`. Es necesario que puedas entrar en la máquina remota sin password para ejecutar una orden remota.
  Sin especificar orden actúa como `rlogin`.

- `rcmd -l nombre unix2 orden ...`
  Actúa como la orden anterior pero además:
  - redirecciona el *stdout* y *stderr* a `/dev/null`
  - la aplicación se ejecuta en *background* remotamente
  - el shell que se arranca remotamente posee las variables `TERM` y `DISPLAY` del shell local (su utilidad es arrancar aplicaciones X-Windows remotas)

- `rcp unix2:/path/file new`
  (remote copy), copia el fichero con camino `/path/file` de la máquina remota `unix2` al fichero `new`. También se puede dar como destino una máquina remota con un fichero local. El orden `rcp` soporta los mismos argumentos de `cp`.

- `telnet unix2`
  (tel network), similar a `rlogin unix2` pero permite specifica el puerta en conexión en la máquina remota.

- `talk usuario1@unix2`
  Intenta hacer una conexión para hablar con el `usuario1` en la máquina `unix2`.
  Existen varias versiones de `talk` en los diferentes sistemas operativos, de forma que no siempre es posible establecer una comunicación entre máquinas con SO's diferentes.
  Existe un comando alternativo: `ytalk`, que pretende ser compatible con todas las versiones de `talk`, permitiendo asimismo la intervención de más de dos personas en la conexión.

[50] AIX de IBM cómo no...

- `ftp unix2`
  (file transfer protocol) aplicación para copiar ficheros entre máquinas de una red. `ftp` exige un nombre de cuenta y password para la máquina remota. Algunas de las opciones más empleadas (una vez establecida la conexión) son:
  - `bin`: Establece el modo de comunicación binario. Es decir, transfiere una imagen exacta del fichero.
  - `asc`: Establece el modo de comunicación ascii. Realiza las conversiones necesarias entre las dos máquinas en comunicación. Es el modo por defecto.
  - `cd`: Cambia directorio en la máquina remoto.
  - `lcd`: Cambia directorio en la máquina local.
  - `ls`: Lista el directorio remoto.
  - `!ls`: Lista el directorio local.
  - `prompt` : No pide confirmación para transferencia múltiple de ficheros.
  - `get rfile [lfile]` : transfiere el fichero `rfile` de la máquina remota a la máquina local denominándolo `lfile`. En caso de no suministrarse el segundo argumento supone igual nombre en ambas máquinas.
  - `send lfile [rfile]` : transfiere el fichero `lfile` de la máquina local a la máquina remota denominándolo `rfile`. En caso de no suministrarse el segundo argumento supone igual nombre en ambas máquinas. También puede usarse `put`.
  - `mget rfile` : igual que `get`, pero con más de un fichero (`rfile` puede contener carácteres comodín)
  - `mput lfile` : igual que `put`, pero con más de un fichero (`lfile` puede contener carácteres comodín).

  Las versiones de `ftp` varían mucho entre las diferentes máquinas y sistemas operativos. Para más información ver páginas de manual.

10.2 E-Mail

El correo electrónico (*E-mail*) es un servicio para el envio de mensajes entre usuarios, tanto de la misma máquina como de diferentes máquinas. Existen multitud de aplicaciones que permiten el uso de este servicio, dentro de los que destacamos:

- `mail`
- `elm`
- `mailtool`
- `xmailtool`

10.2.1 Direcciones de mail

Para mandar un E-mail es necesario conocer la dirección de dicha persona. Esta dirección consta de dos campos que se combinan intercalando entre ellos el `@`:

`user@domain`

- `user`: es la identificación del usuario (i.e. *login*) en la máquina remota.
- `domain`: es la máquina[51] donde dicha persona recibe correo.

Si el usuario es local no es necesario colocar el campo `domain` (ni tampoco el `@`).

10.2.2 Nomenclatura

Veamos algunos conceptos relacionados con el correo electrónico:

- **Subject** : Es una parte de un mensaje que piden los programas al comienzo y sirve como título para el mensaje.
- **Cc** (Carbon Copy): Permite el envío de copias del mensaje que está siendo editado a terceras personas.
- **Reply** : Cuando se envía un mensaje en respuesta a otro se suele añadir el comienzo del subject: `Re:`, con objeto de orientar al destinatario sobre el tema que se responde. Es frecuente que se incluya el mensaje al que se responde para facilitar al destinatario la comprensión de la respuesta.
- **Forward** : Permite el envío de un mensaje (con modificaciones o sin ellas) a una tercera persona.
- **Forwarding Mail** : Permite a un usuario que disponga de cuentas en varias máquinas no relacionadas, de concentrar su correo en una cuenta única[52]. Para ello basta con tener un fichero `$HOME/.forward` que contenga la dirección donde desea centralizar su correo.
- **Mail group** : Un grupo de correo es un conjunto de usuarios que reciben el correo dirigido a su grupo. Existen órdenes para responder a un determinado correo recibido por esa vía de forma que el resto del grupo sepa lo que ha respondido un miembro del mismo.
- **In-Box** : Es el fichero donde se almacena el correo que todavía no ha sido leído por el usuario. Suele estar localizado en `/var/spool/mail/$USER` .
- **Folder** (carpeta): Es un fichero que contiene un conjunto de mensajes. Suele ser una buena costumbre disponer de diferentes folders para las distintas personas o temas...
- **Mailer-Daemon** : Cuando existe un problema en la transmisión de un mensaje se recibe un mensaje proviniente del `Mailer-Daemon` que indica el problema que se ha presentado.
- **8bit transfer & uuencode-uudecode** : Hoy día, casi todos los sitios pueden recibir correctamente correo de 8bit (p.e. letras con tildes). Pero muchos sistemas mantienen límites de tamaño de líneas y/o mensajes. Entonces para mandar ficheros binarios (ejecutables, datos, imagánes,...), existen las ordenes `uuencode` y `uudecode` cuya

[51] A menudo es frecuente que si una persona tiene acceso a un conjunto de máquinas, su dirección de correo no corresponda con una máquina...

[52] Este comando debe usarse con conocimiento pues en caso contrario podría provocar un bucle indefinido y no recibir nunca correo ...

función es convertir el fichero a transmitir en otro empleando sólo 7bits ASCII y líneas de longitud acotada a 62 caracteres. De esta forma se garantiza una transmisión sin problemas.

10.2.3 aplicación `mail`

Es posiblemente la aplicación más simple. Para la lectura de mail teclear simplemente:
`mail`
y a continuación aparece un índice con los diferentes mensajes recibidos. Cada mensaje tiene una línea de identificación con número de orden. Para leer un mensaje basta teclear su número y a continuación RETURN.
Para enviar un mensaje:
`mail ADDRESS`
se pregunta por el `Subject:` y a continuación se introduce el mensaje. Para acabar se teclea sólo un punto en una línea o bien `Ctr-D`. Por último se pregunta por `Cc:` (Carbon Copy).
Es posible personalizar el funcionamiento mediante el fichero `$HOME/.mailrc`.
Para enviar un fichero de texto a través del correo se suele emplear la redirección de entrada:
`mail ADDRESS < FILE`

10.2.4 aplicación `elm`

`elm` es un comando interactivo para el acceso al correo electrónico. Una vez dentro del programa, se muestran las líneas de mensaje:

```
              Folder is 'mbox' with 5 messages [ELM 2.4 PL24]


->    1   Sep 30 Carmen Lopez - Sun (37)   Mas cambios
      2   Oct 4  Carmen Lopez - Sun (32)   Re: Mas cambios
      3   Oct 6  Carmen Lopez - Sun (57)   Mas cambios
      5   Jul 6  aramos@Spain.Sun.C (48)   1a OFERTA


 |=pipe, !=shell, ?=help, <n>=set current to n, /=search pattern
 a)lias, C)opy, c)hange folder, d)elete, e)dit, f)orward, g)roup reply, m)ail,
 n)ext, o)ptions, p)rint, q)uit, r)eply, s)ave, t)ag, u)ndelete, or e(x)it

Command:
```

Las teclas de uso más frecuente son:

- **Ctrl-N** Movimiento al mensaje siguiente
- **Ctrl-P** Movimiento al mensaje anterior
- **Return** Lectura del mensaje apuntado
- **d** Borrado del mensaje seleccionado
- **n** Lectura del mensaje apuntado y avanzar al siguiente
- **r** Responder al mensaje actual
- **g** Responder al grupo que ha enviado el mensaje actual

- **m** enviar un mensaje
- **f** enviar el mensaje actual a una tercera persona
- **q** salir de `elm`

Existe una ayuda en línea al pulsar la tecla **?** . Es posible la personalización modificando el fichero `$HOME/.elm/elmrc`

10.3 News

Qué es Usenet

Usenet es un conjunto de máquinas que intercambian **artículos** asociados a uno o más identificadores denominados **newsgroups**.

Diversidad de Usenet

La definición dada anteriormente puede dar la impresión de algo poco definido. Pero es que no es posible la generalización dado que dentro de Usenet están: agencias gubernamentales, universidades, empresas en general, etc. Cada administrador controla su distribución local, pero nadie gobierna por encima de su dominio...

Usenet no es una democracia

Como Usenet no es una organización y no existe una autoridad central, es difícil que exista una democracia. Aunque pueda parecer contradictorio, para la creación de un nuevo grupo, se procede a una **votación** con objeto de determinar el apoyo que pueda tener la creación de dicho grupo...
Existen grupos locales, es decir, cuya distribución no es mundial.

Estos **newsgroups** abarcan los temas más diversos. Los nombres de los **newsgroups** están formados de forma jerárquica, así por ejemplo: `comp.lang.c` es un grupo sobre computación, particularizando en lenguajes, y más concretamente en lenguaje C. Las siete categorías (campo más significativo del grupo) principales son:

- **comp**: *Topics of interest to both computer professionals and hobbyists, including topics in computer science, software sources, and information on hardware and software systems.*
- **misc**: *Group addressing themes not easily classified into any of the other headings or which incorporate themes from multiple categories. Subjects include fitness, job-hunting, law, and investments.*
- **sci**: *Discussions marked by special knowledge relating to research in or application of the established sciences.*
- **soc**: *Groups primarily addressing social issues and socializing. Included are discussions related to many different world cultures.*
- **talk**: *Groups largely debate-oriented and tending to feature long discussions without resolution and without appreciable amounts of generally useful information.*
- **news**: *Groups concerned with the news network, group maintenance, and software.*

- **rec**: *Groups oriented towards hobbies and recreational activities*
- **alt**: *True anarchy; anything and everything can and does appear; subjects include sex, the Simpsons, and privacy.*

Son muchas las aplicaciones que permiten la lectura de News:

- `rn` : Read News
- `trn` : Threaded Read News.
  Permite el seguimiento de las respuestas a un artículo a través de su árbol de respuesta. De esta forma se entiende de una forma más clara la discusión que se origina.
- `strn` : Scanning Threaded Read News.
  Es una versión extendida de `trn`
- `xrn` : Interface X-Windows para `rn`
- `xvnews` : Interface Xview para `rn`
- `slrn` : Aplicación en S-Lang para leer noticias

10.3.1 aplicación `rn`

Cuando se arranca la aplicación se pueden leer todos los newsgroups a los que uno está substrito. Los comandos más empleados son:

- **Space**: ejecutar acción por defecto, normalmente la acción deseada[53]
- **h**: Help
- **q**: quit el artículo/grupo actual o bien la aplicación según el nivel en el que nos hallemos.
- **n**: siguiente página
- **p**: página anterior
- **g** `newsgroup`: suscribir al newsgroup mencionado
- **u**: desuscribir del newsgroup actual
- **c** (catch up): marcar todos los artículos como leídos
- **k** (kill): marcar todos los artículos con el mismo subject como leídos.

Todos estos comandos son comunes a `trn` y `strn`.

[53] `rn` acierta normalmente lo que quieres hacer...

10.3.2 aplicación `slrn`

Es una de las aplicaciones más cómodas y sencillas de usar desde un terminal de texto, inspirada en el lector *GNUS* de Emacs.
Básicamente dispone de dos modos:

- **Modo de selección de grupos**: En caso de estar suscrito a algún grupo, si se coloca el cursor sobre un grupo y se pulsa el `Return` se cambia de modo y se pasa a leer dicho grupo.
  - `q`: salir de slrn
  - `L`: permite la búsqueda de un grupo que contenga una cadena que se introduce interactivamente a continuación.
  - `s`: suscribir al grupo donde esté el cursor
  - `u`: de-suscribir del grupo donde esté el cursor
  - `p`: comenzar un artículo
- **Modo de lectura de un grupo**: En este modo la pantalla se divide en dos partes: en la superior se hayan los nombres de los artículos y en la inferior se encuentran sus correspondientes contenidos. Los cursores permiten moverse por los diferentes artículos y al pulsar `Return` se lee el artículo señalado.
  - `q`: salir al modo de selección de grupo
  - `Space`: siguiente página del artículo
  - `Del`: página previa del artículo
  - `f`: continuar con el artículo donde esté el cursor (*follow-up*)
  - `w`: almacenar el artículo en un fichero
  - `u`: marcar como no leído

Dispone de una ayuda en línea accesible mediante la tecla `?`.

10.4 ftp Anonymous

Existen nodos que permiten el acceso por **ftp** a usuarios que no disponen de cuenta en dichas máquinas. Para ello se emplea como *login* de entrada el usuario `anonymous` (o `ftp`) y como *passwd* la dirección de E-mail personal. Existen servidores que no aceptan conexiones desde máquinas que no estén dadas de alta en el servicio de nombre (DNS), así como algunas que no permiten la entrada a usuarios que no se identifican correctamente. Dada la sobrecarga que existe, muchos de los servidores tienen limitado el número de usuarios que pueden acceder simultáneamente.

10.5 Archie

Este servicio[54] se creó con objeto de realizar una búsqueda de ficheros por los servidores de ftp anonymous de todo el mundo. Es conveniente conocer el nombre del fichero buscado,

[54]creado por un grupo de la Universidad McGill de Canada

pues la cantidad de ficheros que pueden resultar de una búsqueda "poco específica" es enorme[55].
Para acceder en modo interactivo a dicho servicio uno debe conectarse a un servidor de archie empleando como login `archie`.
Una vez establecida la sesión existe una serie de comandos para ejecutar la búsqueda:

- `quit`: Abandonar la sesión.
- `help` *subject*: ayuda sobre *subject*
- `set` : Modifica las variables de entorno. Algunas de las mas empleadas son:
  - `search`: Establece el modo de la búsqueda
    - `sub`: subcadena (*case insensitive*)
    - `subcase`: subcadena (*case sensitive*)
    - `exact`: cadena (*case sensitive*). El más rápido para cuando se sabe lo que se busca...
    - `regex`: permite la búsqueda de subcadenas empleando *regular expressions*:
      - ^ Subcadena al comienzo
        (Ejemplo)
        `^gs261`
        Busca ficheros que comiencen por `gs261`
      - $ Subcadena al final
        (Ejemplo)
        `gif$`
        Busca ficheros que terminen en `gif`
      - `.*` cualquier subcadena
        (Ejemplo)
        `gs.*tgz`
        Busca ficheros que contengan las subcadenas `gs` y `tgz` (y en ese orden dentro de la cadena)
  - `pager`: habilita el paginador
- `prog` *expression*: permite buscar en la base de datos una determinada cadena. La interpretación de dicha cadena depende del valor de la variable search.
- `mail` *Mail-Address*: permite que el resultado de la búsqueda se envíe por correo a la dirección especificada.

En la actualidad existen diferentes formas de acceder al servicio archie[56]:

- mediante `telnet interactivo`, que es el anteriormente comentado.
- mediante clientes de Xwindows (`xarchie`).
- mediante peticiones por E-mail

[55] En la actualidad hay del orden de unos 800 servidores FTP anonymous, alrededor del mundo. Conteniendo más de 1 millón de ficheros diferentes (cerca de 50 Gbytes de información). Los servidores se suelen actualizar una vez al mes.
[56] `archie.rediris.es` es un ejemplo de un servidor de Archie en España.

10.6 WWW

WWW son las siglas de *World-Wide Web* Este servicio permite el acceso a información entrelazada (dispone de un texto donde un término puede conducir a otro texto): **hyperlinks**. Los ficheros están realizados en un lenguaje denominado **html**. Para acceder a este servicio es necesario disponer de un **lector** de dicho lenguaje. Destacan actualmente:

- `lynx`: lector en modo texto Lynx es un visor en modo texto de la WWW. Eso significa que podrás acceder a cualquier documento, pero sin disfrutar de las imágenes que lo acompañan. Aún así, es un visor muy potente que te permitirá usar *forms* (impresos), escribir correo o leer las news, por ejemplo.

  El funcionamiento es muy sencillo. Veamos algunas de las teclas:

  - `q`: salir de lynx
  - Flecha arriba, flecha abajo: Desplaza el cursor por los diferentes enlaces del documento.
  - Flecha derecha: Avanza un enlace, te lleva hacia donde indique el enlace.
  - Flecha izquierda: vuelve atrás un enlace[57].
  - Tecla de borrado (backspace): Muestra la historia de las páginas que has visitado hasta ahora, y te permite acceder a cualquiera de ellas con rapidez.
  - `g`: Te permite saltar directamente a una dirección de WWW.
  - Tecla 'k': Muestra una guía que te enseña la utilidad de cada tecla.
  - `h` (*Help*): Imagina para qué sirve. :-)
  - `i`: Recupera una página con direcciones potencialmente interesantes.
  - `v`: Te enseña tus *bookmarks*[58].
  - `a`: Añade un *bookmark* a tu archivo personal. Nótese la diferencia con Netscape o Mosaic. Mientras que estos apuntaban la pagina actual, Lynx apunta la dirección que señala el cursor.
  - `r`: Quita un *bookmark* de tu archivo.

- `netscape`: lector en modo gráfico de uso muy extendido, que soporta su propio conjunto de instrucciones html
- `Mosaic`
- `arena`

11 Impresión

Cuando se quiere obtener una copia impresa de un fichero se emplea el comando `lpr`.
`lpr file` – Envía el fichero *file* a la cola de impresión por defecto. Si la cola está activada,

[57] "Desanda" el camino que has recorrido

[58] Piensa en la WWW como si fuera una gran biblioteca. En ciertas ocasiones querrás recordar dónde estaban ciertas páginas que te interesan. Para eso sirve un 'bookmark file'.

la impresora lista y ningún trabajo por encima del enviado, nuestro trabajo será procesado de forma automática.

A menudo existen varias posibles impresoras a las que poder enviar los trabajos. Para seleccionar una impresora en concreto (en vez de la de por defecto) se emplea el modificador:

`lpr -P`*impresora*. Siendo *impresora* el nombre lógico asignado a esta otra impresora[59].

Otras órdenes para la manipulación de la cola de impresión son:

- `lpq [-P`*printer*`]` – Permite examinar el estado de una determinada cola (para ver la cantidad de trabajos sin procesar de ésta por ejemplo).
- `lprm [-P`*printer*`]` *jobnumber* – Permite eliminar un trabajo de la cola de impresión

Uno de los lenguajes de impresión gráfica más extendidos en la actualidad es *PostScript*[60]. Por ello muchas de las impresoras actuales sólo admiten la impresión en dicho formato. En caso de desear imprimir un fichero en ascii deberá previamente realizarse la conversión a PostScript empleando la orden `mpage`:

```
mpage -2 file.txt | lpr
```

Esta orden envía a la impresora el fichero ascii *file.txt* formateado a 2 páginas por hoja.
Hay muchas versiones de UNIX y no todas son iguales. Es posible que en las máquinas encuentres que algunas órdenes no existen, no se han instalado o tienen opciones distintas. Los nombres de impresoras dependen de la instalación, pedir siempre cuáles están disponibles y qué formato de entrada aceptan.
Un fichero PostScript puede ser visualizado antes de imprimirse mediante los comandos:

- `gs [file].ps` : abre una ventana con el contenido de cada página del documento (no es posible retroceder en las páginas)
- `ghostview [file].ps` : análogo al anterior, pero permite diversas opciones (ampliación de una zona, redimensionamiento de la ventana, diversos formatos de papel, etc) así como la posibilidad de recorrer el documento en ambas direcciones.

12 Compresión

A menudo es frecuente la necesidad de crear un fichero comprimido debido al excesivo tamaño de éste, o bien crear un backup de una determinada estructura de directorios para almacenarlos en diskettes. Se comentan a continuación una serie de comandos que permiten ejecutar dichas acciones:

1. `compress [file]`: comprime el fichero, creando el fichero `[file].Z`
2. `uncompress [file.Z]`: descomprime el fichero, creando el fichero `[file]`[61]
3. `zcat [file].Z`: muestra por el *stdout* el contenido descomprimido del fichero (sin destruir el original).

[59] Para recibir una lista de las posibles impresoras de un sistema así como su estado se puede emplear la orden `lpc status all`

[60] PostScript es un lenguaje que permite importación/exportación de figuras entre diferentes aplicaciónes.

[61] Tanto esta orden como la anterior, destruyen el fichero original

Estos compresores son los estándars en UNIX, pero alternativamente pueden usarse:
`gzip/gunzip` : compresor/descompresor de GNU que proporciona un mejor ratio de compresión que compress[62]. La extensión empleada es `[file].gz` o `[file].z`
En caso que se desee crear un fichero comprimido con una estructura de directorios debe ejecutarse la orden:
`tar cvf - [directorio] | compress > [directorio].tar.Z`
o bien:
`tar cvf - [directorio] | gzip > [directorio].tgz`
Y para descomprimir:
`zcat [directorio].tar.Z | tar xvf -`
o bien:
`gunzip -c [directorio].tar.Z | tar xvf -`
Para ver el contenido del fichero comprimido:
`zcat [directorio].tar.Z | tar tvf - | more`
o bien:
`gunzip -c [directorio].tar.Z | tar tvf - | more`
Por comodidad se han definido unos *alias* de estas secuencias de comandos:

- `pack [directorio]`: crea el fichero `[directorio].tgz` que almacena la estructura de directorios debajo del directorio `[directorio]` comprimido empleando el compresor `gzip`.
- `unpack [directorio].tgz`: crea la estructura de directorios contenida en el fichero `[directorio].tgz`
- `packinfo [directorio].tgz`: crea un fichero `[directorio].tgz.memo` que contiene la información sobre los archivos contenidos en el fichero comprimido.

Existe un comando análogo a `more` denominado `less` (`m`), que permite visualizar (paginando) el contenido de un fichero, incluso si éste se haya comprimido.
De todas estas órdenes existen páginas de manual disponibles por si es necesario consultar las opciones disponibles.

13 Compilación y Debugging

13.1 cc & gcc

El comando para usar el compilador de lenguaje C es `cc`. Su uso más elemental es:
`cc FILENAME.c`
que compila el fichero `FILENAME.c` y crea un fichero ejecutable que se denomina `a.out` por omisión.
Existen diversas opciones que comentaremos a continuación:

- **-c** realiza la compilación pero no el link:
  `cc -c FILENAME.c`
  genera el fichero `FILENAME.o` que es código objeto.

[62] Los ficheros no son compatibles con `pkzip/pkunzip` de MSDOS, existe `zip/unzip` que sí lo son

- **-o EXENAME** define el nombre del ejecutable creado (en lugar del defecto `a.out`):
  `cc -o OUTPUTFILE FILENAME.c`
- **-lx** incluye una librería en la compilación:
  `cc FILENAME.c -lm`
  En este caso se compila con la librería matemática (libm.a).
- **-g** permite el uso de un debugger
- **-O** optimización

Otro compilador generalmente disponible es `gcc` (the GNU C compiler). Su uso es aproximadamente el mismo que el de `cc` y con las mismas opciones de éste. La principal diferencia es que `gcc` es compatible ANSI (mientras que `cc` solo soporta la versión Kernighan & Ritchie). Una de las opciones de que dispone `gcc` es:
`-Wall` que detecta posibles errores/warnings en el código C que está siendo compilado.

13.2 `make` & `Makefile`

Frecuentemente los programas están compuestos por diferentes subrutinas que se hayan contenidas en diferentes ficheros. La orden de compilación necesaria puede ser engorrosa, y a menudo no es necesario volver a compilar todos, los ficheros, sino sólo aquellos que hayan sido modificados.
UNIX dispone de una orden denominada `make` que evita los problemas antes mencionados y permite el mantenimiento de una librería personal de rutinas. Este comando analiza qué ficheros de código han sido modificados después de la última compilación y evita recompilaciones innecesarias.
En su uso más simple sólo es necesario suministrar una lista de dependencias y/o instrucciones a la orden `make` en un fichero denominado `Makefile`. Una dependencia es la relación entre dos ficheros de forma que un fichero se considera actualizado siempre que el otro tenga una fecha de modificación inferior a éste.
Por ejemplo si el fichero `file.c` incluye el fichero `file.h`, no se puede considerar actualizado el fichero `file.o` si el fichero `file.c` o el fichero `file.h` ha sido modificado después de la última compilación. Se dice que el fichero `file.o` depende de `file.c` y el fichero `file.c` depende de fichero `file.h`.
La sintaxis para establecer una dependencia es:
`FILE1: DEP1 DEP2 ...`
        instrucciones para generar FILE1
`FILE2: DEP3 DEP4 ...`
        instrucciones para generar FILE2
Las instrucciones *deben de estar indentadas por un tabulador*[63].
Por ejemplo un fichero Makefile podría tener una apariencia como:

```
file.o: file.c file.h
	cc -c file.c
```

En este caso se comprueba las fechas de las última modificaciónes de los fichero `file.c` y `file.h`, si estas fecha son más recientes que la del fichero `file.o` se procede a la compilación.

[63] ¡ocho espacios *no* es igual a un tabulador para `make`!

El comando `make` se puede suministrar con un argumento, que indica la etiqueta situada a la izquierda de los dos puntos. Así en el ejemplo anterior podría invocarse `make file.o`.
`make` tiene un mecanismo para la creación del `Makefile`. Los macros se definen al comienzo con una sintaxix:
`MACRO1=` definición macro1
`MACRO2=` definición macro2
Una macro puede ser utilizada en el resto del `Makefile` colocando un `$` delante de él. Por defecto `make` sabe las órdenes y dependencias (reglas) para compilar un fichero `*.c` y producir un fichero `*.o`, entonces basta especificar solamente los dependecias que `make` no pueda adivinar de los nombres de los ficheros, p.e.:

```
OUTPUTFILE   = prog
OBJS         = prog.o misc.o aux.o
INCLUDESMISC = misc.h aux.h
INCLUDESFILE = foo.h $(INCLUDESMISC)
LIBS         = -lmylib

prog.o: $(INCLUDESFILE)
misc.o: $(INCLUDESMISC)
aux.o: aux.h

$(OUTPUTFILE): $(OBJS)
        cc $(OBJS) -o $(OUTPUTFILE) $(LIBS)
```

13.3 `dbx` debugger

`dbx` es una utilidad para depuración de errores (*debugging*) de programas escritos en C, Pascal y Fortran 77. Permite la localización de problemas en un programa al permitir una ejecución paso a paso (o bien indicando puntos de parada) y la observación de las variables del programa.
Para emplear `dbx` es necesaria la compilación con la opción `-g` que genera la tabla de símbolos necesaria. La ejecución del debugger se realiza mediante:
`dbx FILENAME`
Siendo `FILENAME` el nombre del ejecutable deseado. Veamos a continuación algunos de los comandos más empleados:

- `run`
  comienza/continua la ejecución hasta donde se haya definido la parada (o hasta el fin de la ejecución si no se han definido condiciones de parada)

- `rerun`
  comienza la ejecución desde el principio

- `stop at LINENUMBER`
  detiene la ejecución en la línea `LINENUMBER`

- `stop in PROCEDURE`
  detiene la ejecución cuando se alcance la rutina `PROCEDURE`

- `stop VARIABLE`
  detiene la ejecución cuando se modifique el valor de la variable `VARIABLE`
- `step N`
  ejecuta las `N` siguientes líneas (una sóla si no se proporciona argumento), entrando en las funciones
- `next N`
  ejecuta las `N` siguientes líneas (una sóla si no se proporciona argumento), sin entrar en las funciones
- `list FIRTS, LAST`
  muestra las líneas comprendidas entre `FIRST` y `LAST`
- `print VARIABLE`
  muestra el valor de `VARIABLE` en el punto actual de ejecución
- `assign VARIABLE=VALUE`
  asigna a `VARIABLE` el valor `VALUE` en el punto actual de ejecución
- `whatis VARIABLE`
  muestra la declaración de `VARIABLE`
- `where N`
  muestra las `N` funciones activas en la pila (es muy útil cuando se ha producido un `core`, esta orden muestra las funciones que han sido llamadas cuando se ha producido)
- `quit`
  salida de `dbx`

También pueden emplearse condicionales `if` (con sintaxis de C) para los comandos `step`. Por ejemplo:
`stop at 120 if a==42`
indica que debe detener la ejecución en la línea 120 si la variable `a` tiene el valor de 42 en este punto.
Es posible la definición de alias. Por ejemplo:
`alias s step`
Estos alias constituyen una personalización y se suelen almacenar en el fichero `$HOME/.dbxinit`, que es leído al comienzo de la ejecución de `dbx`.
Existen otros debuggers que citaremos a continuación:

- `gdb` (GNU's debugger) es un debugger muy potente sólo para programas en C. Funciona de forma análoga a `dbx`, pero añade una serie de órdenes adicionales.
- `xgdb` versión X-Windows del anterior.

14 FAQ (Frecuently Asked Questions)

Colección de preguntas que está prohibido hacer:

1. **He borrado un fichero sin copia**:
   Si borras un fichero por error, y no tienes un 'backup' ¡¡es imposible recuperarlo!!

2. **Qué significa el mensaje XXXX ?**:

   - `Error: Can't open display:`
     Falta ejecutar la orden :
     `setenv DISPLAY [hostname]:0`
   - `Xlib: connection to "[hostname]:0.0" refused by server`
     `Xlib: Internal error during connection authorization check`
     `Error: Can't open display: [hostname]:0`
     Falta la orden:
     `xhost + [client_hostname]`
   - `vmunix: NFS server not responding still trying`
     El disco duro importado por la máquina no está actualmente accesible por red, debido a un problema de comunicaciones (normalmente alguien ha abierto la red en algún sitio ¡¡sin avisar!!). La solución es esperar...
   - `csh% hspice rc.cir > rc.out`
     `rc.out: file exists`
     Se está intentando redireccionar sobre un fichero que ya existe, una solución (si se quiere sobreescribir es):
     `csh% hspice rc.cir >! rc.out`
     o bien cambiar el nombre del fichero:
     `csh% hspice rc.cir > rc.out2`
   - `csh%/usr/local`
     `/usr/local: Permission denied.`
     No es que se te prohiba ejecutar algún programa, es que LOS DIRECTORIOS NO SE EJECUTAN.
   - `csh% csh% man tcsh`
     `csh%: Command not found.`
     Cuando en este manual se pone `csh%`, se quiere hace referencia al prompt del sistema, PERO NO HAY QUE TECLEARLO.

15 Reference Charts

15.1 UNIX Reference

Dal MSCS Unix Reference Card

Logging In

Type your **username** to the system login prompt.
Type your **password** to the system password prompt.

Logging Out

Type **logout**, or **exit**.

Change Password

Type **passwd**, and answer the system prompts for old password, new password, and verification of new password.

Help

help on a certain subject	`man -k` *subject*
help on a certain command	`man` *command*
online information	`info`

Files

The characters `?` and `*` are used for pattern matching. A `?` matches any single character, and a `*` matches an arbitrary number of characters.

Creating

create file	`see` *Editing*

Removing/Renaming/Moving/Copying

remove file	`rm` *filename*
rename file	`mv` *filename newname*
move file to another directory	`mv` *filename dirname*
copy file to another filename	`cp` *filename newfile*
copy file to another directory	`cp` *filename dirname*

Listing

list file(s)	`ls` *filename(s)*
list files sorted by time of last modification	`ls -t` *filenames*

Directories

Like filenames, directories use `?` and `*` as pattern matching characters (see *Files*). The current directory and parent directory, are named `.` and `..` respectively.

MSCS Dalhousie University, v1.0

Creating/Traversing

create/make a directory	`mkdir` *dirname*
change directory to home directory	`cd`
change directory to parent directory	`cd ..`
change directory to another directory	`cd` *dirname*
print working directory	`pwd`

Removing/Renaming/Moving/Copying

remove empty directory	`rmdir` *dirname*
remove directory and contents	`rm -r` *dirname*
rename directory, or move directory to another directory	`mv` *dirname newdir*
copy directory and contents to another directory	`cp -r` *dirname dir*

Listing

list current directory contents	`ls`
list directory contents	`ls` *dirname*
list directory contents sorted by time of last modification	`ls -t` *dirname*

I/O Redirection

The standard input/output/error of a command, normally associated with the terminal, may be redirected by appending one the following to the command line:

open file as standard input	`<`*filename*
open file as standard output	`>`*filename*
open file as standard output and standard error	`>&`*filename*
append standard output to file	`>>`*filename*
append standard output and standard error to file	`>>&`*filename*
pipe standard output of command1 to standard input of command2	*cmd1* | *cmd2*

Editing

create/edit file - visual editor	`vi` *filename*
create/edit file - emacs editor	`emacs` *filename*
create/edit file - line editor	`ex` *filename*

Printing

print text file on screen	`more` *filename*
print text file on line printer	`lpr` *filename*
print text file on laser printer	`lw` *filename*
print tex document on laser printer	`lw -d` *filename*
print troff document on laser printer	`lwtroff` *filename*

Formatting Documents

format a tex document	`tex filename`
format a latex document	`latex filename`
format and print a troff document	`lwtroff filename`
format a nroff document	`nroff filename`

TeX and laTeX files must have **.tex** as a filename extension.
TeX and laTeX produce files with **.dvi** as a filename extension.

Electronic Mail

read mail	`mail`
send mail to some **address**	`mail` *address*
mail a file to some **address**	`mail` *address* `<`*file*

When sending mail; type your message and then enter a `.` or `<ctrl>d` on a line by itself to send it.

News

read news	`rn`
post news	`Pnews`

Languages and Compilers

compile C lanuage source file	`cc` *filename*
compile Fortarn lanuage source file	`f77` *filename*
compile Pascal lanuage source file	`pc` *filename*
run Common Lisp interpreter	`cl`
Awk pattern scanning and processing language	`awk -f` *programfile*
S statistical programming language	`S`

C, Fortran, and Pascal source files must have the extensions **.c**, **.f**, and **.p** respectively.

Commands

calulator	`bc`
monthly calendar	`cal` *month year*
yearly calendar	`cal` *year*
current date and time	`date`
show differences between two files	`diff` *file1 file2*
display information about a user	`finger` *username*
file transfers to/from a remote host	`ftp` *hostname*
find lines matching a specific **string**	`grep` *string file(s)*
find lines not matching a specific **string**	`grep -v` *string file(s)*
show the first num lines of filename	`head -`*num filename*
view online manual pages	`man` *command*
login to a remote unix host	`rlogin` *hostname*
execute a command on a remote unix host	`rsh` *host command*
spelling checker	`spell` *filename*
show the last num lines of filename	`tail -`*num filename*
talk to another user	`talk` *username*
connect to a remote host	`telnet` *hostname*
display information on the top CPU processes	`top`
show how long the system has been up	`uptime`
display number of lines, words, and character in a file	`wc` *filename*
show who is on the system	`who`

Department of Mathematics, Statistics, & Computing Science
Dalhousie University
November 1989 v1.0

15.2 ELM Reference

Elm Reference Card

Starting Elm

read mail from system mailbox	elm
read mail from a folder	elm -f =*folder*
read mail from a file	elm -f *filename*

Leaving Elm

move to the character in front of the quit, maybe prompting for deleting, storing, and keeping messages	q
quick quit, no prompting	Q
exit, leaving folder untouched if changed	x
exit, leaving folder untouched if changed	<ctrl>q
exit, leaving folder untouched unconditionally	X

Motion commands

display next index page	+
display next index page	→
display previous index page	-
display previous index page	←
set current message to first message	=
set current message to last message	*
set current message to message <NUMBER>	<NUMBER><RETURN>
increment current message by one	j
increment current message by one	<ctrl>n
increment current message by one	↓
advance to next undeleted message	J
advance to next undeleted message	<esc>n
decrement current message by one	k
decrement current message by one	<ctrl>p
decrement current message by one	↑
advance to previous undeleted message	K
advance to previous undeleted message	<esc>p

Display commands

display current message	<RETURN>
display current message	<SPACE>
display current message with headers	h
redraw screen	<ctrl>l

MSCS Dalhousie University, v1.0

Message handling commands

pipe current message or tagged messages to a system command	|
search 'From'/'Subject' headers for a pattern	/
search entire message for a pattern	//
save current message or tagged messages to a folder	>
save current message or tagged messages to a folder	s
scan current message for calendar entries	<
bounce (remail) current message	b
copy current message or tagged messages to a folder	C
delete current message and decrement by one	D
delete current message and increment by one	d
delete messages with a specified pattern	<ctrl>d
forward current message	f
group reply to current message	g
limit messages by specified criteria	l
mail a new message	m
next message, display current then increment by one	n
print current message or tagged messages	p
reply to current message	r
tag current message	t
tag messages with a specified pattern	<ctrl>t
undelete current message and decrement by one	U
undelete current message and increment by one	u
undelete messages with a specified pattern	<ctrl>u

Other commands

execute a shell command	!
resyncronize folder	$
help	?
change to alias mode	a
change folder	c
change ELM options mode	o

Department of Mathematics, Statistics, & Computing Science
Dalhousie University
June 1990 v1.0

GNU Emacs Reference Card

(for version 18)

Starting Emacs

To enter Emacs, just type its name: `emacs`

To read in a file to edit, see Files, below.

Leaving Emacs

suspend Emacs (the usual way of leaving it)	`C-z`
exit Emacs permanently	`C-x C-c`

Files

read a file into Emacs	`C-x C-f`
save a file back to disk	`C-x C-s`
insert contents of another file into this buffer	`C-x i`
replace this file with the file you really want	`C-x C-v`
write buffer to a specified file	`C-x C-w`
run Dired, the directory editor	`C-x d`

Getting Help

The Help system is simple. Type `C-h` and follow the directions. If you are a first-time user, type `C-h t` for a **tutorial**. (This card assumes you know the tutorial.)

get rid of Help window	`C-x 1`
scroll Help window	`ESC C-v`
apropos: show commands matching a string	`C-h a`
show the function a key runs	`C-h c`
describe a function	`C-h f`
get mode-specific information	`C-h m`

Error Recovery

abort partially typed or executing command	`C-g`
recover a file lost by a system crash	`M-x recover-file`
undo an unwanted change	`C-x u` or `C-_`
restore a buffer to its original contents	`M-x revert-buffer`
redraw garbaged screen	`C-l`

Incremental Search

search forward	`C-s`
search backward	`C-r`
regular expression search	`C-M-s`

Use `C-s` or `C-r` again to repeat the search in either direction.

exit incremental search	`ESC`
undo effect of last character	`DEL`
abort current search	`C-g`

If Emacs is still searching, `C-g` will cancel the part of the search not done, otherwise it aborts the entire search.

Motion

Cursor motion:

entity to move over	**backward**	**forward**
character	`C-b`	`C-f`
word	`M-b`	`M-f`
line	`C-p`	`C-n`
go to line beginning (or end)	`C-a`	`C-e`
sentence	`M-a`	`M-e`
paragraph	`M-[`	`M-]`
page	`C-x [`	`C-x ]`
sexp	`C-M-b`	`C-M-f`
function	`C-M-a`	`C-M-e`
go to buffer beginning (or end)	`M-<`	`M->`

Screen motion:

scroll to next screen	`C-v`
scroll to previous screen	`M-v`
scroll left	`C-x <`
scroll right	`C-x >`

Killing and Deleting

entity to kill	**backward**	**forward**
character (delete, not kill)	`DEL`	`C-d`
word	`M-DEL`	`M-d`
line (to end of)	`M-0 C-k`	`C-k`
sentence	`C-x DEL`	`M-k`
sexp	`M-- C-M-k`	`C-M-k`

kill **region**	`C-w`
kill to next occurrence of *char*	`M-z` *char*
yank back last thing killed	`C-y`
replace last yank with previous kill	`M-y`

Marking

set mark here	`C-@` or `C-SPC`
exchange point and mark	`C-x C-x`
set mark *arg* **words** away	`M-@`
mark **paragraph**	`M-h`
mark **page**	`C-x C-p`
mark **sexp**	`C-M-@`
mark **function**	`C-M-h`
mark entire **buffer**	`C-x h`

Query Replace

interactively replace a text string	`M-%`
using regular expressions	`M-x query-replace-regexp`

Valid responses in query-replace mode are

replace this one, go on to next	`SPC`
replace this one, don't move	`,`
skip to next without replacing	`DEL`
replace all remaining matches	`!`
back up to the previous match	`^`
exit query-replace	`ESC`
enter recursive edit (`C-M-c` to exit)	`C-r`

Multiple Windows

delete all other windows	`C-x 1`
delete this window	`C-x 0`
split window in 2 vertically	`C-x 2`
split window in 2 horizontally	`C-x 5`
scroll other window	`C-M-v`
switch cursor to another window	`C-x o`
shrink window shorter	`M-x shrink-window`
grow window taller	`C-x ^`
shrink window narrower	`C-x {`
grow window wider	`C-x }`
select a buffer in other window	`C-x 4 b`
find file in other window	`C-x 4 f`
compose mail in other window	`C-x 4 m`
run Dired in other window	`C-x 4 d`
find tag in other window	`C-x 4 .`

Formatting

indent current **line** (mode-dependent)	`TAB`
indent **region** (mode-dependent)	`C-M-\`
indent **sexp** (mode-dependent)	`C-M-q`
indent region rigidly *arg* columns	`C-x TAB`
insert newline after point	`C-o`
move rest of line vertically down	`C-M-o`
delete blank lines around point	`C-x C-o`
delete all whitespace around point	`M-\`
put exactly one space at point	`M-SPC`
fill **paragraph**	`M-q`
fill **region**	`M-g`
set fill column	`C-x f`
set prefix each line starts with	`C-x .`

Case Change

uppercase word	`M-u`
lowercase word	`M-l`
capitalize word	`M-c`
uppercase region	`C-x C-u`
lowercase region	`C-x C-l`
capitalize region	`M-x capitalize-region`

The Minibuffer

The following keys are defined in the minibuffer.

complete as much as possible	`TAB`
complete up to one word	`SPC`
complete and execute	`RET`
show possible completions	`?`
abort command	`C-g`

Type `C-x ESC` to edit and repeat the last command that used the minibuffer. The following keys are then defined.

previous minibuffer command	`M-p`
next minibuffer command	`M-n`

GNU Emacs Reference Card

Buffers

select another buffer	`C-x b`
list all buffers	`C-x C-b`
kill a buffer	`C-x k`

Transposing

transpose **characters**	`C-t`
transpose **words**	`M-t`
transpose **lines**	`C-x C-t`
transpose **sexps**	`C-M-t`

Spelling Check

check spelling of current word	`M-$`
check spelling of all words in region	`M-x spell-region`
check spelling of entire buffer	`M-x spell-buffer`

Tags

find tag	`M-.`
find next occurrence of tag	`C-u M-.`
specify a new tags file	`M-x visit-tags-table`
regexp search on all files in tags table	`M-x tags-search`
query replace on all the files	`M-x tags-query-replace`
continue last tags search or query-replace	`M-,`

Shells

execute a shell command	`M-!`
run a shell command on the region	`M-|`
filter region through a shell command	`C-u M-|`
start a shell in window `*shell*`	`M-x shell`

Rmail

scroll forward	`SPC`
scroll reverse	`DEL`
beginning of message	`.` (dot)
next non-deleted message	`n`
previous non-deleted message	`p`
next message	`M-n`
previous message	`M-p`
delete message	`d`
delete message and back up	`C-d`
undelete message	`u`
reply to message	`r`
forward message to someone	`f`
send mail	`m`
get newly arrived mail	`g`
quit Rmail	`q`
output message to another Rmail file	`o`
output message in Unix-mail style	`C-o`
show summary of headers	`h`

Regular Expressions

The following have special meaning inside a regular expression.

any single character	`.` (dot)
zero or more repeats	`*`
one or more repeats	`+`
zero or one repeat	`?`
any character in set	`[ ... ]`
any character not in set	`[^ ... ]`
beginning of line	`^`
end of line	`$`
quote a special character *c*	`\`*c*
alternative ("or")	`\\|`
grouping	`\( ... \)`
*n*th group	`\`*n*
beginning of buffer	``\` ``
end of buffer	`\'`
word break	`\b`
not beginning or end of word	`\B`
beginning of word	`\<`
end of word	`\>`
any word-syntax character	`\w`
any non-word-syntax character	`\W`
character with syntax *c*	`\s`*c*
character with syntax not *c*	`\S`*c*

Registers

copy region to register	`C-x x`
insert register contents	`C-x g`
save point in register	`C-x /`
move point to saved location	`C-x j`

Info

enter the Info documentation reader	`C-h i`

Moving within a node:

scroll forward	`SPC`
scroll reverse	`DEL`
beginning of node	`.` (dot)

Moving between nodes:

next node	`n`
previous node	`p`
move **up**	`u`
select menu item by name	`m`
select *n*th menu item by number (1–5)	*n*
follow cross reference (return with `l`)	`f`
return to last node you saw	`l`
return to directory node	`d`
go to any node by name	`g`

Other:

run Info **tutorial**	`h`
list Info commands	`?`
quit Info	`q`
search nodes for regexp	`s`

Keyboard Macros

start defining a keyboard macro	`C-x (`
end keyboard macro definition	`C-x )`
execute last-defined keyboard macro	`C-x e`
append to last keyboard macro	`C-u C-x (`
name last keyboard macro	`M-x name-last-kbd-macro`
insert lisp definition in buffer	`M-x insert-kbd-macro`

Commands Dealing with Emacs Lisp

eval **sexp** before point	`C-x C-e`
eval current **defun**	`C-M-x`
eval **region**	`M-x eval-region`
eval entire **buffer**	`M-x eval-current-buffer`
read and eval minibuffer	`M-ESC`
re-execute last minibuffer command	`C-x ESC`
read and eval Emacs Lisp file	`M-x load-file`
load from standard system directory	`M-x load-library`

Simple Customization

Here are some examples of binding global keys in Emacs Lisp. Note that you cannot say `"\M-#"`; you must say `"\e#"`.

```
(global-set-key "\C-cg" 'goto-line)
(global-set-key "\e\C-r" 'isearch-backward-regexp)
(global-set-key "\e#" 'query-replace-regexp)
```

An example of setting a variable in Emacs Lisp:

```
(setq backup-by-copying-when-linked t)
```

Writing Commands

```
(defun ⟨command-name⟩ (⟨args⟩)
  "⟨documentation⟩"
  (interactive "⟨template⟩")
  ⟨body⟩)
```

An example:

```
(defun this-line-to-top-of-screen (line)
  "Reposition line point is on to the top of
the screen.  With ARG, put point on line ARG.
Negative counts from bottom."
  (interactive "P")
  (recenter (if (null line)
                0
              (prefix-numeric-value line))))
```

The argument to `interactive` is a string specifying how to get the arguments when the function is called interactively. Type `C-h f interactive` for more information.

For copies of the GNU Emacs manual, write to the Free Software Foundation, Inc., 675 Massachusetts Ave, Cambridge MA 02139.

Vi Reference Card

Starting Vi

edit an empty buffer	vi
edit a file	vi filename

Leaving Vi

write buffer to file	:w
write buffer to file **filename**	:w filename
quit	:q
quit discarding changes	:q!
write changes and quit	:wq
write changes and quit	ZZ

Special Notes

Use the escape key, <esc>, to leave insert mode or to cancel an incomplete command.

If a command doesn't work, try hitting <esc> and entering the command again.

Motion commands

move up one line	k
move down one line	j
move left one character	h
move right one character	l
scroll up one line	<ctrl>y
scroll down one line	<ctrl>e
scroll up (default is a half page)	<ctrl>u
scroll down (default is a half page)	<ctrl>d
show next page page	<ctrl>f
show previous page page	<ctrl>b
move to beginning of next line	+
move to beginning of previous line	-
move to beginning of first screen line	H
move to middle line of screen	M
move to last line of screen	L
move to the **n**th line	nG
move to the **n**th line	:n
move to the next occurrence of **c** on the current line	fc
move to the character in front of the next occurrence of **c** on the current line	tc
move forward to the next word	w
move forward to the end of the next word	e
move backward to the previous word	b

All commands except M, nG, and :n can be preceeded by an integer to indicate the distance to move.

Text entering commands

append text at end of line	A
append text after cursor	a
insert text at beginning of line	I
insert text before cursor	i
open a new line above cursor	O
open a new line below cursor	o

All commands must be terminated with an <esc> after the text has been entered.

Text deletion commands

delete character left of cursor	X
delete character under cursor	x
delete rest of word	dw
delete rest of sentence	d)
delete rest of paragraph	d}
delete line containing cursor	dd
delete the current line upto and including **c**	dfc
delete the current line upto **c**	dtc

All commands can be preceeded by and integer to indicate the number of characters, words, lines etc. to be deleted.

Text alteration commands

replace text	R
replace character with **c**	rc
transpose two characters	xp
join the next line to the end of the current line	J
undo last change	u
undo last change to current line	U
change rest of word	cw
change rest of sentence	c)
change rest of paragraph	c}

The commands R, cw, c) and c} must terminated with an <esc> after the new text has been entered. All commands, except xp, u, and U, can be preceeded by an integer to indicate the amount of alteration to perform.

Text moving commands

yank a copy of the current line and place it in a buffer	yy
put the last item yanked or deleted before the cursor	P
put the last item yanked or deleted after the cursor	p
yank a copy of the current line and place it in buffer **c**	"cY
put contents of buffer **c** on a new line above the cursor	"cP
put contents of buffer **c** on a new line below the cursor	"cp

The commands Y and "cY can be preceeded by and integer to indicate the number of lines to copy.

Search Commands

search forward for **pattern**	:/pattern/
search forward for **pattern**	/pattern/
search backward for **pattern**	:?pattern?
search backward for **pattern**	?pattern?
delete the next line containing **pat1**	:/pat1/d
substitute the next occurance of **pat1** with **pat2**	:/pat1/s/pat1/pat2
repeat the last search	n
repeat the last search in opposite direction	N

Global parameter

substitute **pat2** for **pat1**	:g/pat1/s/pat1/pat2
delete all occurrences of **pat1**	:g/pat1/d
print all occurrences of **pat1**	:g/pat1/p

Addresses

the **n**th line	n
lines **n** through **k**	n,k
the current line	.
the last line	$
n lines after the current line	+n
n lines before the current line	-n
the last **n**+1 lines	$-n,$

Instructions

delete lines **n** through **k**	:n,k d
copy lines **n** through **k** after line **j**	:n,k co j
move lines **n** through **k** after line **j**	:n,k mo j
substitute first occurance of **pat1** with **pat2** on the last 8 lines	:$-8,$ s/pat1/pat2/

Miscellaneous

read file **filename** into the buffer after the current line	:r filename
read the output of command **cmd** into the buffer after the current line	:r !cmd
write the buffer to file **filename** replacing its previous contents	:w! filename
write the buffer to the end of file **filename**	:w >> filename
turn on line numbering	:set number
turn off line numbering	:set nonumber
turn on auto indenting	:set autoindent
turn off auto indenting	:set noautoindent

Department of Mathematics, Statistics, & Computing Science
Dalhousie University
November 1989 v1.0

Indice Alfabético