INTRODUCCIÓN BÁSICA AL SISTEMA OPERATIVO UNIX

Introducción Básica al Sistema Operativo UNIX

ÍNDICE

0.- Consideraciones previas

Este guión tiene como fin guiar al alumno en su familiarización con el sistema UNIX. No se pretende realizar un manual completo que describa sus características, funciones y utilidades, sino que sea una guía básica de introducción. Esto implica que muchos puntos de los a continuación expuestos pueden resultar incompletos para el lector y para subsanar este problema le sugerimos que emplee el manual on line del propio UNIX.

Acceso al manual

El manual de UNIX nos ofrece una ayuda on-line que está dividida en ocho secciones:

Sección 1	Comandos de usuario (User Commands).
Sección 2	Llamadas al sistema (System calls).
Sección 3	Biblioteca de funciones de C (C Library Functions).
Sección 4	Dispositivos e interfaces de red (Devices and Network Interfaces).
Sección 5	Formatos de ficheros (File Formats).
Sección 6	Juegos y demostraciones (Games and Demos).
Sección 7	Varios (Miscellaneous).
Sección 8	Referencia del Administrador (Administrator Reference).

El manual se encuentra en /usr/man y se accede a él con el formato:

```
man [sección] comando
```

Las páginas solicitadas se formatean con el programa `nroff` y las muestra con la utilidad `more`. Si no se especifica otra opción se muestra la primera página del manual que se encuentre con ese nombre.

```
man -k palabra
```

consulta el índice en busca de alguna referencia a la palabra especificada como argumento. El resultado se muestra en un listado de todas las entradas del índice que contienen la palabra buscada.

Partes de una entrada del manual:

Name	Nombre y función. Estas líneas forman el índice que se consulta mediante `man -k`
Synopsis	Diagrama sintáctico: el nombre, seguido de las opciones (entre corchetes) y posibles argumentos. En el caso de llamadas al sistema y rutinas de biblioteca, el formato y tipo de los parámetros.
Description	Breve descripción.
Return Value	En entradas correspondientes a llamadas al sistema y rutinas de biblioteca, lista de los posibles valores de retorno.
Errors	En entradas correspondientes a llamadas al sistema, lista de los posibles errores.
Options	En entradas correspondientes a comandos, lista detallada de las posibles opciones y sus efectos.
Commands	En entradas correspondientes a comandos interactivos, lista detallada de los mandatos propios de la utilidad.
Files	Ficheros relacionados.
See Also	Referencias a otras entradas del manual y otros tipos de documentación.
Diagnostics	En entradas correspondientes a comandos, lista de mensajes de diagnóstico y error que pueden producirse.
Bugs	Problemas conocidos, o cuestiones pendientes de resolver.

1.- Introducción

Historia.

La primera versión de UNIX fue desarrollada por Ken Thompson en los Laboratorios Bell (AT&T) en 1969. Se empleó por primera vez sobre una máquina PDP-7 de DEC. Se programó en ensamblador, y cuando Dennis Ritchie desarrolló el lenguaje C, se reescribió en C.

Versiones.

Actualmente se emplean principalmente cuatro versiones diferentes de este sistema operativo, que son:

- UNIX System V distribuido por AT&T.
- BSD v. 4.3 distribuido por la Universidad de California en Berkeley.
- SunOS/Solaris distribuido por la empresa SUN.
- Linux

En 1984 comenzó el desarrollo de un sistema operativo similar a UNIX de libre distribución dentro de lo que se dio en llamar el Proyecto GNU. En la actualidad se emplean ampliamente variaciones del sistema de GNU con el kernel de Linux, llamándose sistema Linux/GNU (http://www.gnu.org).

El MIT (Instituto Tecnológico de Massachusetts) distribuye desde 1984, y de forma gratuita, una interfaz gráfica basada en este sistema operativo que se denomina X-Window.

2.- Funcionamiento inicial

UNIX es un sistema operativo multiusuario, es decir, permite que más de un usuario utilice simultáneamente el sistema. Para hacer esto de forma coherente cada usuario debe identificarse para utilizar el sistema, es decir, se necesita una *cuenta* en el sistema. La cuenta está formada básicamente por un *nombre de usuario* y una *clave de acceso o password.* El usuario deberá introducir su nombre de usuario y su clave. A partir de ahí el sistema lanza un primer programa (generalmente lo que se conoce como una Shell) que se ejecuta con el identificador de ese usuario.

En el laboratorio se cuenta con un cierto número de ordenadores, dedicados para las prácticas de diversas asignaturas, así como para proyectistas. Cada grupo de prácticas dispone de una cuenta con su correspondiente nombre de usuario. Para iniciar la sesión, cada usuario debe introducir por teclado su nombre de usuario ante el mensaje por pantalla: `login`:

La primera vez, no se tendrá la password asignada, así que cuando salga por pantalla: `Password:`

se deberá dar al `ENTER`. Una vez iniciada la sesión, será cada usuario el que elija su propia password, mediante el comando `yppasswd`. La password debe tener al menos 8 caracteres y al menos un carácter debe ser numérico. Es obvio decir que cada password debe ser celosamente guardada por su dueño para evitar que un intruso entre en el sistema identificándose como él.

El sistema de ficheros montado en el laboratorio posibilita que los usuarios no se tengan que poner siempre en el mismo ordenador para trabajar con sus ficheros, por lo tanto puede emplearse cualquier ordenador o cambiar de uno a otro.

Cada usuario tiene asignado un directorio en el sistema de ficheros. Ese directorio consta como propiedad de ese usuario y en él éste tiene permiso para crear / borrar / modificar ficheros o cambiar permisos.

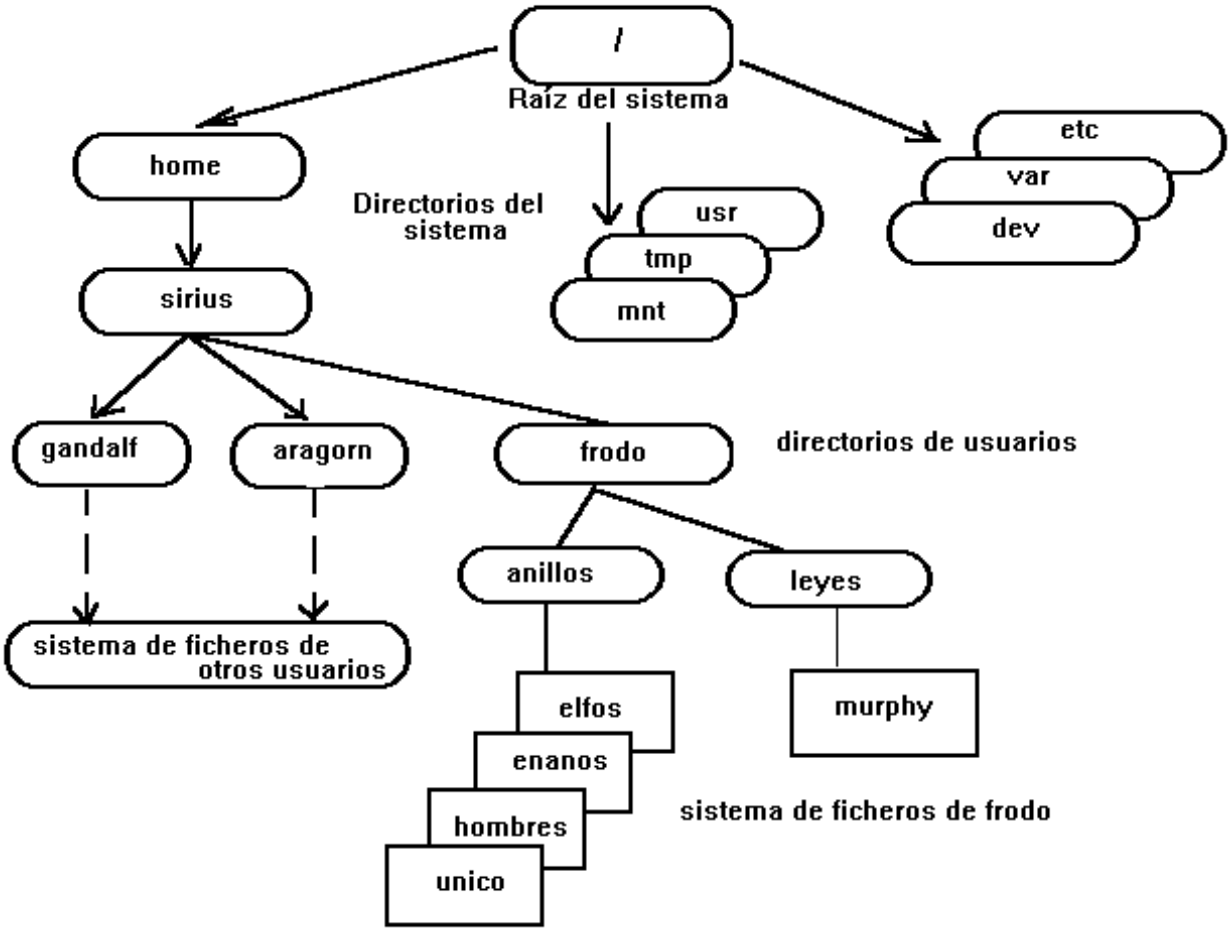
Una vez iniciada la sesión de trabajo se recomienda ejecutar el comando `startx` para iniciar una sesión del entorno de ventanas X-Window. Para concluir la sesión de trabajo hay que salir del entorno de ventanas, y luego, salir de la cuenta mediante el comando `exit`. Debe aparecer de nuevo la petición: `login:`

Nota importante: No apagar los ordenadores, aunque sí los monitores. UNIX es un sistema multiusuario y multitarea como ya se ha comentado, puede haber usuarios utilizando el sistema a través de la red que no desean ver cortado su trabajo.

3.- El sistema de ficheros

Un sistema de ficheros proporciona un métido conveniente para organizar y almacenar ficheros. Todos los ficheros en UNIX residen en un sistema de ficheros, si importar el tipo de los mismos.

El sistema de ficheros en UNIX es una organización de directorios y ficheros estructurada en forma de arbol. La *raíz* de un sistema de ficheros de UNIX se repersenta con el carácter /.



Un fichero puede ser idenificado de forma única especificando la ruta desde la raíz hasta él en el árbol de directorios, en tal caso la ruta empieza en /. Si la ruta no comienza por / se entiende que empieza en el directorio de trabajo del proceso que intenta hacer referencia a ese fichero.

Un fichero consiste en una sucesión de bytes terminada por una marca de *fin de fichero*. Físicamente un fichero puede contener una serie de bloques de disco o cinta. Los bloques de datos que pertenecen a un fichero pueden estar dispuestos de forma aleatoria en el sistema de almacenamiento.

Ficheros ordinarios

Un fichero ordinario contiene datos arbitrarios en cero o más bloques de datos almacenados en un sistema de ficheros. Estos ficheros pueden contener texto ASCII o datos binarios. No existe ninguna estructura impuesta por el sistema operativo sobre cómo se debe organizar un fichero. UNIX no hace ninguna distinción entra ficheros que contienen diferentes tipos de datos.

Directorios

Los directorios son un tipo especial de ficheros que proporcionan la relación entra nombres de ficheros y los ficheros propiamente dichos. Como resultado de esto, la estructura de los directorios define la estructura del sistema de ficheros completo.

Un directorio consiste en una tabla cuyas entradas, una para cada fichero, contienen: un número de inodo y un nombre de fichero empleado para hacer referencia, de forma simbólica, a ese inodo. Cada entrada en la tabla del directorio se emplea para convertir el nombre de un fichero en su correspondiente inodo.

Cada proceso (programa en ejecución) se encuentra siempre en un directorio, es lo que se llama su *directorio de trabajo (working directory)*, que el proceso puede cambiar a voluntad.

Ficheros especiales

Los ficheros especiales no contienen datos. En vez de eso proporcionan un mecanismo para relacionar dispositivos físicos con nombres de fichero en el sistema de ficheros. Cada dispositivo soportado por el sistema está asociado con al menos un fichero especial. Cuando se realiza una petición de lectura o escritura sobre un fichero especial resulta en la activación del controlador asociado con ese dispositivo, este controlador es la parte del código del sistema encargada de controlar las operaciones relacionadas con el dispositivo físico.

Menciamos a continuación utilidades importantes relacionadas con el manejo de ficheros:

• ls

Lista ficheros. Si no se especifica el directorio se toma el directorio detrabajo del proceso que lo ejecuta (algunas opciones útiles: `-alF`).

• `cat`

Muestra por pantalla el contenido de un fichero. Todos los ficheros ordinarios en UNIX son similares, es decir, son simplemente un conjunto de bytes. No hay diferencia entre lo que en otros sistemas se conoce como ficheros de texto y ficheros binarios. Sin embargo, ficheros que no contengan texto simple generalmente contendrán bytes con valores que no hagan referencia a caracteres imprimibles. Por lo tanto no es aconsejable utilizar `cat` (o cualquier otra utilidad para mostrar el contenido de ficheros) sobre ficheros que no son texto simple.

• `cp <origen> <destino>`

Hace una copia de un fichero.

• `mv <origen> <destino>`

Mueve un fichero de un lugar a otro. En realidad elimina del directorio la entrada que hace referencia a ese fichero y añade una nueva en el directorio especificado en el destino. Sirve también para cambiar el nombre que tiene un fichero en un directorio.

• `rm <fichero>`

Elimina la entrada en un directorio referente a un fichero. Con la opcion `-r` elimina tambíen directorios, aunque estos no estén vacíos.

• `mkdir <nombre>`

Crea un nuevo directorio.

• `rmdir <nombre>`

Elimina un directorio. Es necesario que no contenga ningún fichero.

• `pwd`

Muestra el directorio de trabajo actual.

• `cd <nombre>`

Permite cambiar el directorio de trabajo del Shell.

• `chmod <permisos> <fichero>`

Permite cambiar el conjunto de permisos de un fichero.

Permisos de los ficheros

En UNIX todos lo ficheros tienen un propietario y un grupo. El propietario es quien lo creo y el grupo generalmente es el grupo al que pertenece el propietario.

Todos los ficheros tienen unos permisos que permiten a unos usuarios u otros realizar ciertas operaciones con ellos. Los permisos están en tres categorías:

• Permisos de propietario: Son los que se aplican al propietario del fichero.

• Permisos de grupo: Se aplican a todos los miembros de ese grupo que no son el propietario.

• Permisos para el resto: Se aplican a todos los que no entran en ninguna de las dos categorías anteriores.

En cada una de estas categorías hay tres permisos:

• Permiso de lectura: Permite leer el fichero.

• Permiso de escritura: Permite modificar el fichero.

• Permiso de ejecución: Permite ejecuarlo.

En el caso en que el fichero es un directorio el permiso de lectura permite listar su contenido y el de ejecución permite mover el directorio de trabajo de un proceso a ese directorio. Para eliminar un fichero hace falta permiso de escritura en el directorio que lo contiene (se debe modificar la tabla de ese directorio). No es necesario tener permiso de escritura en el fichero.

Estos permisos se organizan de la siguiente forma:

Ejemplo:

```
-rwx-w----
```

El primer caracter es el tipo de fichero:

Tipos:

-	normal
d	directorio
c	dispositivo
s	fichero para comunicación entre procesos (socket)
l	enlace simbólico

Los siguientes 3 caracteres representan los permisos del propietario, luego vienen los del grupo y finalmente los del resto de usuarios.

Los símbolos son los siguientes:

r	leer
w	escribir
x	ejecutar

Como ya se ha mencionado, la utilidad `chmod` sirve para cambiar los permisos de un fichero. Tiene fundamentalmente dos modos de empleo:

Ejemplo 1:

```
% chmod 754 fichero
```

Entiende una sintaxis numérica, a cada categoría le asigna un dígito octal, de tal modo que 1 permite el acceso, y 0 no lo permite. El primer dígito representa los permisos del propietario. 7 en binario es 111, lo cual corresponde a los tres permisos activos (lectura, escritura, ejecución). El segundo dígito representa los permisos del grupo. 5 en binario es 101, lo cual corresponde a lectura y ejecución activado, escritura desactivado. El tercer dígito son los permisos para el resto del mundo. 4 en binario es 100, solo permiso de lectura.

Ejemplo 2:

```
% chmod o+r fichero
```

En este caso se especifica mediante una letra qué permisos se desea modificar:

- Clase: u : propietario
  g : grupo
  o : resto
  a : todos

Después se especifica la operación que se desea realizar sobre el permiso:

- Operación: + : añade acceso
  - : elimina acceso
  = : pone permiso

Y a continuación sobre qué permiso se desea actuar:

- Permiso: r : lectura
  w : escritura
  x : ejecución

El ejemplo añadiría permiso de lectura al resto de usuarios.

4.- El C-Shell

Cuando nos autentificamos ante el sistema, éste lanza un proceso que ejecuta un primer programa para nosotros. Generalmente este programa es lo que se conoce como una Shell o intérprete de comandos. Es un programa que se dedica a recoger del teclado instrucciones respecto a comandos que deseamos ejecutar y a ejecutarlos. Puede añadir muchas otras facilidades.

El Shell que vamos a comentar es el C-Shell (csh). Este intérprete de mandatos fue programado en lenguaje C en la Universidad de California (Berkeley, EEUU). Otro Shell que está bastante difundido es el Bourne Shell (sh), que tiene su origen en los Laboratorios Bell y que distribuye AT&T. El C-Shell es más sofisticado que el Bourne Shell, pero este último es más rápido. Estas características hacen que el C-Shell sea óptimo para un uso interactivo, y que el segundo se emplee para escribir rutinas de mandatos (scripts).

Todo lo que se describe a continuación hace referencia al C-Shell.

En un mandato el orden es el siguiente:

```
% comando opción(es) argumento(s) [redireccionamiento(s)]
```

Cuando el Shell ejecuta un mandato le asigna una entrada estándar, una salida estándar y una salida de error estándar. Normalmente la entrada es el teclado y las salidas se ofrecen en pantalla. Cuando un programa lee de la entrada estándar y escribe en la salida estándar se dice que dicho programa es un filtro.

El C-Shell permite redirigir los tres canales estándar (entrada, salida y error) a ficheros. La expresión utilizada para redirigir la salida estándar es la siguiente:

```
comando > fichero
```

Ejemplo:

```
% ls -alF > listado
```

Crea el fichero; lo vacía antes si ya existía. Si lo que se desea es que la salida del comando se añada al contenido de un fichero basta con formar el comando de la siguiente manera:

```
comando >> fichero
```

Ejemplo:

```
% echo Fin del listado >> listado
```

Se puede hacer que la entrada del comando sea el contenido de un fichero:

```
comando < fichero
```

Ejemplo:

```
% cat < listado
```

También permite conectar la salida estándar de un comando con la entrada estándar de otro; esto es lo que se denomina una *pipe*. Se pueden especificar varias pipelines en una sola línea. Para indicar al Shell que se desea hacer esta conexión se emplea el carácter `|`.

Ejemplo:

```
ls | wc -l
```

`ls` lista los ficheros de un directorio y envía mediante la pipe esa salida a la utilidad `wc` que con la opción `-l` cuenta el número de líneas.

Comandos incluidos en el Shell

- `cd` Permite cambiar el directorio de trabajo del Shell
- `echo args` Muestra sus argumentos por la salida estándar.

Variables

Los Shells soportan dos tipos de variables: variables locales y variables de entorno. Ambos tipos de variables almacenan datos en forma de una cadena. La diferencia principal entre ambos es que cuando el Shell crea otro Shell (ejecutando `/bin/csh`), el hijo tiene una copia de las variables de entorno del padre, pero no de las locales.

Cada Shell tiene un conjunto de variables de entorno predefinidas, generalmente en ficheros de inicialización, así como variables locales.

Para acceder al contenido de las variables del Shell hay que colocar `$` delante del nombre de las mismas.

Comandos del C-Shell relacionados con las variables:

- `set` Lista las variables definidas.
- `set var = cont` Permite crear la variable `var` y darle por valor `cont`.
- `unset var` Destruye la variable `var`.
- `echo $var` Muestra el contenido de la variable `var`.
- `printenv` Lista las variables de entorno definidas
- `setenv var cont` Permite crear la variable `var` y darle por valor `cont`.
- `unsetenv var` Destruye la variable `var`.

Variables de entorno comunes:

Nombre	Significado
HOME	El directorio del usuario, el camino completo.
PATH	Lista de directorios donde el Shell buscará los comandos que se le pida ejecutar. Si no los encuentra ahí devuelve error.
USER	Identificador del usuario.
SHELL	Shell en uso, el camino completo.
TERM	Tipo de terminal en uso.

Se puede emplear símbolos para expandir nombres de ficheros, de modo que se simplifique el manejo de éstos. Algunos son:

*****	Cualquier carácter
?	Un único carácter
[car1...carn]	Cualquier carácter de la lista o rangos incluidos entre los corchetes.
{cadena,...}	Cada una de las cadenas de la lista.
~usuario	Directorio inicial del usuario especificado. Si no se especifica el usuario, sino solo el ~ (Altgr+ñ), se entiende que hace referencia al directorio del usuario propietario del proceso Shell.

Historia de comandos

Podemos mantener una historia de eventos que nos permita emplearlos sin tener que teclearlos de nuevo. Para ello hay que crear una variable de entorno del Shell y darle como valor el número de eventos (comandos) que queremos que recuerde (que serán los últimos). Esta variable se llama `history`. Existe un comando de igual nombre (`history`) que lista todos los eventos almacenados. Si deseamos repetir uno de esos eventos tenemos varias posibilidades; algunas de las más típicas se listan a continuación:

• `history`	Presenta el listado de eventos.
• `set history =` nº_de_eventos	Establece el número máximo de eventos que serán conservados.
• `!!`	Evento previo.
• `!n`	Evento n-ésimo.
• `!cad`	Evento más reciente que comience con la cadena `cad`.

Ficheros del Shell

El Shell puede ser personalizado por el usuario. Se configura por medio de tres ficheros:

- `.cshrc`
- `.login`
- `.logout`

Como se puede observar los tres ficheros comienzan con un punto, por lo tanto si queremos listarlos deberemos emplear la opción `-a` en el comando `ls`.

El Shell lee al comienzo el fichero `.cshrc` y lo ejecuta, luego hace lo mismo con el fichero `.login`. Al finalizar la sesión lee y ejecuta `.logout`. Tanto `.login` como `.logout` sólo se ejecutan una vez, aunque después se activen otros shells estos ficheros no se vuelven a ejecutar.

6.- Caracteres especiales

Algunos caracteres son interpretados de forma especial al ser tecleados en un terminal. Suelen llamarse metacaracteres y se pueden listar con la utilidad stty (`stty -a`)

Ejemplo de parte del resultado:

```
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>; eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
```

El ^ ante una letra significa que se ha de pulsar la tecla *Ctrl* al mismo tiempo que esa letra. El significado de algunos de estos caracteres es:

- intr Termina la ejecución de un proceso.
- eof Su significado es *Fin de Fichero* , en los casos en que se emplea la entrada estándar para dar información a un comando sirve para enviar este carácter que da por finalizada la información.
- erase Corresponde al carácter que provoca el borrado de la letra enterior a la posición del punto de inserción.

7.- Utilidades generales de UNIX

El número de utilidades existentes para UNIX nos obliga a hacer un resumen muy básico.

`grep`	Busca una cadena en uno o más ficheros (o en la entrada estándar) Ej. `grep <cadena> <fichero>`
`head`	Permite ver las primeras líneas de un fichero (o de la entrada estándar). Sin opciones muestra las 10 primeras.
`tail`	Permite ver las últimas líneas de un fichero (o de la entrada estándar). Sin opciones muestra las 10 últimas.
`more`	Permite la observación pausada de un fichero (o de la entrada estándar). Pulsando la barra espaciadora avanza una página, con `b` o `^B` retrocede una página, con `q` termina.
`wc`	Cuenta el número de bytes, palabras o líneas en un fichero (o en la entrada estándar).
`who`	Muestra los usuarios que en este momento están trabajando con el sistema.
`df`	Permite saber cuánto espacio libre hay en cada sistema de ficheros.
`du`	Dice cuántos bloques ocupa un directorio con todos sus ficheros y subdirectorios.
`find`	Realiza una búsqueda recursiva, comenzando por el directorio especificado y descendiendo por los subdirectorios. Ej.: `find / -name mifichero -print` Buscaría desde el directorio raíz ficheros con el nombre `mifichero` y sacaría los resultados por pantalla.

Editor `vi`

Es un programa editor de ficheros de pantalla que no necesita el entorno de ventanas X-Window. Tiene varios modos de funcionamiento, entre los cuales cabe destacar el modo de inserción de texto y el modo de comandos. Para pasar del modo de inserción al modo de comandos se presiona la tecla *Esc*.

Destacaremos las funciones más básicas del modo de comandos:

a	Añadir texto (pasa al modo de inserción y todo lo que tecleemos se añadirá al fichero) tras el carácter sobre el que se halla el cursor.
i	Pasa al modo de inserción e inserta texto delante del carácter sobre el que se encuentra el cursor.
ZZ	Graba el fichero y sale del editor.
:x	Graba el fichero y sale del editor.
:w	Graba el fichero sin salir del editor.
:w fichero	Graba en el fichero con el nombre indicado.
:q	abandona el editor.
:q!	Abandona el editor sin grabar.
x	Elimina el carácter sobre el que está el cursor.
dd	Elimina la línea sobre la que está el cursor.
numdd	Elimina las num líneas situadas tras el cursor.
dw	elimina la palabra sobre la que está el cursor.
u	Deshace la última operación.

Utilidades de red

Estas utilidades nos permiten compartir los recursos de la red. Nuestra máquina puede ver incrementada su potencia y versatilidad gracias a los recursos de otra máquina que pertenezca a nuestra misma red.

`telnet` Permite acceder a otros sistemas. El formato es:

```
telnet hostname
```

Seguramente la máquina nos responderá con su petición de `login:`

`ftp` Permite la transferencia de ficheros entre sistemas en red aunque tengan distintos sistemas operativos. El formato de inicio es: `ftp hostname`. Con `get` nos traemos un fichero de una máquina remota y con `put` llevamos un fichero a la máquina remota. Con `help` obtenemos una lista de los comandos disponibles en ese servidor.

Conclusión

Se recomienda explorar el sistema, en especial la información disponible mediante la utilidad `man`. Existen numerosos libros sobre UNIX que pueden resultar de gran utilidad a la hora de familiarizarse con el sistema, se recomienda acudir a ellos.

Bibliografía

UNIX For Programmers And Users A Complete Guide, G. Glass, Ed. Prentice Hall, ISBN 0-13-061771-7

Advanced Programming In The UNIX Environment, W. Richard Stevens, Ed. Addison-Wesley, ISBN 0-201-56617-7

Beggining Linux Programming, N.Matthew & R.Stones, Ed.Wrox, ISBN 1-874416-68-0