1. Sistema de ayuda

UNIX dispone de forma estándar de un completo sistema de ayuda. Podemos obtener ayuda sobre cualquier comando o sobre cualquier aspecto del sistema mediante el comando *man*, cuyo formato es:

```
   man [sección] materia
```
o
```
man -k clave
```
donde:

materia es el elemento (comando, llamada al sistema, etc.) sobre el cual se solicita información

sección es el capítulo del manual en el que se busca la información sobre la materia en cuestión. Este argumento es opcional, y en caso de no especificarse, se busca información acerca de la materia seleccionada en todos los capítulos del manual, mostrándose la primera información que se encuentre.

La opción *–k* seguida de un argumento permite buscar información mediante una palabra clave. Si nosotros necesitamos información sobre un comando cuyo nombre no recordamos, pero sabemos algo de lo que hace, probamos a buscar información mediante una palabra clave, mostrándonos las páginas de ayuda de todos los tópicos en cuya página aparezca la palabra clave que hemos especificado.

Ejemplo 1 *Supongamos que deseamos encontrar ayuda sobre el compilador de C del sistema, pero que no nos acordamos de cómo se llama. En ese caso, ejecutaríamos el siguiente comando, para pedir ayuda al sistema sobre todo aquello en cuya página aparezca la palabra " compiler":* man -k compiler

Otra posibilidad de ayuda que tienen la mayoría de los sistemas UNIX es a través del propio comando. Cuando un comando se invoca con argumentos no válidos (bien sea por ser incorrectos, o por ser insuficientes) el mismo comando nos muestra un breve forma de uso (usage) del mismo. Por ejemplo, si ejecutamos el siguiente comando cp en el que faltan los dos argumentos para el comando cp, el sistema responderá con un mensaje del tipo:

```
Uso:  cp [-hip] [--] src destino
o:  cp [-hip] [--] src1 ... srcN directorio
o:  cp {-R | -r} [-hip] [--] dir1 ... dirN dir_destino
```

No obstante, este mecanismo no está estandarizado, por lo que la salida producida en este caso puede variar de un sistema a otro. En algunos sistemas, por ejemplo, los comandos admiten una opción --help para mostrar ayuda sobre sí mismos.

2. El intérprete de comandos

El intérprete de comandos es el programa que recibe lo que se escribe en el terminal y lo convierte en instrucciones para el sistema operativo. En otras palabras, el objetivo de cualquier intérprete de comandos es ejecutar los programas que el

usuario teclea en el *prompt* del mismo. El *prompt* es una indicación que muestra el intérprete para anunciar que espera una orden del usuario. Cuando el usuario escribe una orden, el intérprete ejecuta dicha orden. En dicha orden, puede haber programas internos o externos: los programas internos son aquellos que vienen incorporados en el propio intérprete, mientras que los externos son programas separados.

En el mundo Linux/UNIX existen tres grandes familias de *Shells* como se muestra en la figura 1. Estas se diferencian entre si básicamente en la sintaxis de sus comandos y en la interacción con el usuario.

2.1. Sintaxis de los comandos

Los comandos tienen la siguiente sintaxis: *programa* arg_1 arg_2 ... arg_n. Se observa que, en la "l´ınea de coman- dos", se introduce el programa seguido de uno o varios argumentos. Así, el intérprete ejecutará el programa con las opciones que se hayan escrito.

Tipo de Shell	Shell estándar	Clones libres
AT&T Bourne shell	sh	ash, bash, bash2
Berkeley "C" shell	csh	tcsh
AT&T Korn shell	ksh	pdksh, zsh
Otros intérpretes	–	esh, gush, nwsh

Figura 1: Intérpretes de comandos en Linux/UNIX

(\). Además, cuando se quiere ejecutar varios comandos en la misma línea, los separa con punto y coma (;). Cuando se quiere que el comando sea de varias líneas, se separa cada línea con el carácter barra invertida ejemplo:

```
# make modules ; make modules install
```

En los comandos también se pueden utilizar los comodines:

- El asterisco (*) es equivalente a uno o más caracteres en el nombre de un archivo. Ej: ls *.tex lista todos los archivos que terminan en ".tex".
- El signo de interrogación (?) es equivalente a un único carácter. Ej: ls boletin.te? lista el archivo boletin.tex completando el último carácter.
- Un conjunto de caracteres entre corchetes es equivalente a cualquier carácter del conjunto. Ej: ls cursolinux.t[aeiou]x lista cursolinux.tex seleccionando la e del conjunto.
- Si inmediatamente después de la apertura del corchete el primer carácter es "^" entonces el significado de la selección se invierte, es decir, representa a todos los caracteres que no están en la lista o en el intervalo. Ej: ls cursolinux.t[^iou]x no seleccionará a: cursolinux.tix, cursolinux.tox, cursolinux.tux

2.2. Redireccionamiento de E/S

La filosofía de Linux/UNIX es en extremo modular. Se prefieren las herramientas pequeñas con tareas puntuales a las meta-herramientas que realizan todo. Para hacer el modelo completo es necesario proveer el medio para ensamblar estas herramientas en estructuras más complejas. Esto se realiza por medio del redireccionamiento de las entradas y las salidas.

Todos los programas tienen por defecto una entrada estándar (teclado) y dos salidas: la salida estándar (pantalla) y la salida de error (pantalla). En ellos se puede sustituir la entrada y salidas estándar por otro dispositivo utilizando los caracteres "<" y ">", es decir, hacer que se lea un archivo que contenga las opciones a ejecutar y un archivo de salida, respectivamente.

Si por ejemplo se quisiera saber los archivos que empiezan por *i* o *I* y almacenarlo en un archivo el comando

ls [iI]* > listado.txt sería suficiente.

Es importante resaltar que el carácter de redirección de salida ">" destruirá el archivo al cual apunta, si este existe, para ser reemplazado por uno nuevo con los resultados del proceso. Si se desea anexar la información a uno ya existente debe usarse doble carácter ">>".

2.3. Tuberías o *pipes*

En la línea de comandos la integración entre diferentes programas se realiza por medio de la redirección de las entradas y salidas a través de *pipes* o tuberías.

Una tubería o pipe es una combinación de varios comandos que se ejecutan simultáneamente, donde el resultado del primero se envía a la entrada del siguiente. Esta tarea se realiza por medio del carácter barra vertical "|". Por ejemplo, si se quieren ver todos los archivos que hay en el directorio /usr/bin, se ejecuta lo siguiente:

ls /usr/bin | more

De este modo, la salida del programa ls (listado de todos los archivos del directorio /usr/bin) irá al programa

more(modo paginado, es decir, muestra una pantalla y espera a que pulsemos una tecla para mostrar la siguiente).

Dentro de esta estructura se han construido una serie de programas conocidos como "filtros" los cuales realizan procesos básicos sobre textos (ver tabla 3).

Filtros	Función
sort	Ordena las líneas de un texto
cut	Corta secciones de una línea
od	Convierte archivos a forma octal u otras
paste	Une líneas de diferentes archivos
tac	Concatena e imprime archivos invertidos
tr	Traduce o borra caracteres
uniq	Elimina líneas repetidas
wc	Cuenta bytes, palabras y líneas

Figura 3: Algunos Filtros en línea de comandos Linux/UNIX

3. Comandos básicos de UNIX

3.1. Comandos para el manejo de ficheros

ls El comando *ls* lista un conjunto de ficheros, el contenido de un directorio, el contenido de un árbol de directorios o cualquier combinación de los anteriores. Su formato es: *ls [opciones] nombre*. El formato del listado lo establecen las opciones. Algunas de las más usuales son:

-l muestra un listado largo, que contiene información detallada de los ficheros.

-a lista todos los ficheros, incluyendo aquellos cuyo nombre comienza por el carácter '.'.

-R lista los directorios de forma recurrente

-t lista en orden cronológico, comenzando por los más recientemente actualizados.

cp/mv Los comandos *cp* y *mv* se emplean respectivamente para copiar y mover ficheros, o incluso para copiar subárboles de directorios en el caso de *cp*. El formato de ambos comandos es: *cp/mv [opciones] origen*$_1$ *[origen*$_2$ *... origen*$_n$*] destino*, donde *origen*$_i$ son los ficheros, conjuntos de ficheros especificados mediante comodines, o directorios que se copian o mueven. Cuando se copian o mueven múltiples ficheros, el destino debe ser obligatoriamente un directorio. *destino* es el nombre de fichero destino o el directorio al que se copia o se mueve. Las opciones más comunes son:

-f No avisar si la operación machaca ficheros destino.

-i Avisar y pedir confirmación si la operación machaca ficheros destino.

-u No copiar ni mover ficheros que sobreescriban a ficheros de igual nombre con fecha de última modificación igual o posterior a la de los mismos.

-r Copiar subdirectorios de forma recurrente (sólo *cp*).

chmod El comando *chmod* se usa para seleccionar autorizaciones de acceso a un archivo o directorio. Es posible asignar tres clases de autorización: Leer (indicado por una r"), escribir (indicado por una "w") y ejecutar (válido únicamente para programas; indicado por una "x").

Hay tres grupos de personas a los que se puede otorgar cada una de las autorizaciones (leer, escribir y ejecutar): el propietario del archivo o directorio (conocido como Usuario), el grupo al que pertenece el propietario (conocido como Grupo) y todos los demás (Otros)

El siguiente es el formato básico: *chmod grupos[+|−]permisos fichero*

Por ejemplo, este comando:
chmod o+x editor.pl

otorga a todos los demás (Otros) autorización para ejecutar el archivo editor.pl (un script en perl). Este comando:

chmod go-w mydata.dat

quita (el signo menos) el permiso de escribir (w) de los conjuntos

de usuarios Grupo y Otros. También se pueden representar

permisos en formato octal, es decir:

r = 4 w = 2 x = 1

rwx = 7 (4+2+1)
rw- = 6 (4+2)
r-x = 5 (4+1)

rw-r--r-- = 644
rw ------ = 600
rwxr-xr-x = 755

Para cambiar los permisos para que sólo el propietario pueda leerlo y escribirlo, teclee:

chmod 600 <fichero>

Para que además sea ejecutable por todos:

chmod 755 <fichero>

3.2. Comandos para el manejo de la Entrada/Salida

cat El comando *cat* escribe el contenido de uno o más ficheros de texto en la salida estándar. Su formato es: *cat [opciones] [fichero*$_1$ *fichero*$_2$ *fichero*$_n$*]*, donde *fichero*$_i$ son los ficheros cuyos contenidos se escriben en

la salida estándar. En caso de que no se especifique ningún fichero, o que se especifique el carácter '–' como nombre de fichero, *cat* escribe su entrada estándar sobre la salida estándar. Algunas de las opciones más frecuentes son:

-b Enumera todas las líneas que no estén en blanco, a partir de 1.

-n Enumera todas las líneas, tanto las que están en blanco como las que no.

more El comando *more* permite visualizar el contenido de un fichero de texto página a página. Normalmente este comando es utilizado por otros comandos o por terceras aplicaciones para visualizar su salida. Ejemplo de comando que hace esto suele ser *man*.El formato del comando more es: *more [opciones] fichero*$_1$ *[fichero*$_2$ . . . *fichero*$_n$*]*, donde *fichero*$_i$ son los ficheros cuyos contenidos se muestran página a página. Las opciones
más comunes son:

-n donde n es el número de líneas que se muestran por cada página.

-f hace que *more* cuente líneas lógicas en lugar de físicas. Esto evita que las líneas largas se muestren usando varias l´ıneas en pantalla, forzando a que se muestren truncadas.

-p suprime el *scroll*. En su lugar, por cada página limpia la pantalla y muestra el texto a continuación.

-c suprime el *scroll*. En su lugar, por cada página comienza escribiendo en la primera línea de la pantalla, y a continuación escribe las líneas de texto, borrando la porción de cada línea de pantalla que no se use.

-s compacta varias l´ıneas en blanco consecutivas en una sola l´ınea en blanco.

+n donde n es un número. Comienza en la línea n–ésima.

+/patrón busca la primera ocurrencia en el texto del patrón, comenzando en dicho punto la presentación.

echo Los comandos *echo* y *print* muestran en la salida estándar una cadena dada, entendiendo una cadena como una secuencia de palabras separadas por caracteres de tabulación o espacios en blanco. Tras la cadena mos- trada se produce un salto de l´ınea. El formato de ambos comandos es: *echo cadena*, *print [-n] cadena*, donde
-n indica que no se debe producir el salto de línea a continuación de la cadena.

read El comando *read* lee de la entrada estándar el valor de una o más variables. El formato del comando es: *read variable*$_1$ *[variable*$_2$ . . . *variable*$_n$*]*, donde variable$_i$ son los nombres de las variables que se leen. El comando *read* lee una l´ınea completa de texto, asignando una palabra a cada variable. Las palabras se supone que están delimitadas por tabuladores o espacios en blanco. En caso de que se lean más palabras que
variables, todas las palabras "de sobra" al final de la línea se asignaran a la última variable. Si el número de palabras es menor que el número de variables, las últimas variables reciben como valor una cadena vacía.

grep El comando *grep* toma como entrada uno o más ficheros, y muestra en la salida estándar aquellas líneas de los ficheros de entrada en la que se encuentre una subcadena que cumpla un patrón dado. Si se especifican múltiples ficheros de entrada, cada línea de salida va precedida por el nombre del fichero. Si no se especifica un fichero de entrada, o si se especifica el carácter '-' como nombre de fichero, *grep* lee de la entrada estándar.
El formato del comando *grep* es: *grep [opciones] patrón [fichero*$_1$ *fichero*$_2$ … *fichero*$_n$*]*, donde fichero$_i$ son los ficheros cuyas líneas se procesan y patrón es el patrón que se busca. Este puede ser una expresión regular

de la forma que se van a describir a continuación. Es una buena costumbre encerrar el patrón entre comillas simples. Por defecto, interpreta el patrón como una expresión regular básica. Las opciones más comunes son:

-E Interpreta el patrón como una expresión regular extendida.

-F Interpreta el patrón como una o más cadenas fijas, separadas por caracteres de nueva línea.

-h Suprime el nombre de fichero al principio de cada línea aun en el caso de que se procesen múltiples ficheros.

-i No distingue entre mayúsculas y minúsculas.

-l Muestra sólo una lista con los ficheros de la entrada que en algún lugar contienen el patrón.

-v Hace que *grep* muestre las líneas que no contienen el patrón.

-w Requiere que el patrón coincida con una palabra completa

-f f Indica a *grep* que lea la expresión regular del fichero *f* en lugar de la línea de comandos

Una expresión regular es una plantilla de texto construida mediante caracteres literales y alguno(s) de los metacaracteres siguientes, y cuya finalidad es representar a un conjunto de cadenas. Si una cadena puede ser representada mediante la expresión regular, se dice que la cadena "satisface"dicha expresión.

Los metacaracteres con los que podemos escribir las expresiones regulares básicas en UNIX son:

Metacarácter	Significado
.	Representa a cualquier carácter
[lista de caracteres] o [carácter$_1$–carácter$_n$]	Representa a uno cualquiera de los caracteres de la lista, o a cualquier carácter com- prendido entre carácter$_1$ y carácter$_n$ según el orden ASCII. Si el primer carácter tras el corchete [es el carácter '^', el significado se invierte, es decir, representa a todos los caracteres que no están en la lista o en el intervalo. Dentro de los corchetes, los metacaracteres '$', '*', y '/' pierden su significado especial.
*	Pospuesta a cualquier expresión y significa cero o más ocurrencias de dicha expresión.
^	Antepuesta a cualquier expresión regular, indica que la expresión debe aparecer al comienzo de la l´ınea solamente.
$	Pospuesta a cualquier expresión regular, indica que la expresión debe aparecer al final de la l´ınea solamente.
\	El significado de cualquier metacarácter puede ser ignorado antecediéndole por la barra inversa ('\'), en cuyo caso el metacarácter se interpreta de forma literal.

Figura 4: Expresiones regulares.

who El comando *who* proporciona información sobre los usuarios conectados a la máquina. Su formato es: *who [opciones]*. Si es invocado sin opciones, proporciona la siguiente información por cada usuario conectado en el momento: Nombre de usuario, dispositivo lógico (tty) al que está conectado, tiempo que lleva conectado (normalmente, fecha y hora de conexión), nombre de la máquina o *display* X desde el que se conecta. Las opciones más comunes son:

-m Igual que 'who am i'

-q Proporciona el nombre de los usuarios conectados e indica cuántos hay en total.

-u Tras la hora de conexión, muestra el tiempo (horas y minutos) que el usuario lleva inactivo. Un punto ('.') indica que el usuario ha estado activo en el último minuto, y la cadena 'old' indica que el usuario lleva más de 24 horas inactivo.

sort El comando *sort* se emplea para ordenar, fusionar ordenadamente o comprobar si están ordenadas todas las líneas del fichero o ficheros de entrada. Por defecto, *sort* escribe en la salida estándar. Su formato es: *sort [opciones] [fichero$_1$ fichero$_2$ . . . fichero$_n$]*, donde *fichero$_i$* son los ficheros de entrada. Si no se especifica fichero de entrada, o si se especifica '-' como fichero de entrada, *sort* leerá de la entrada estándar.

El comando *sort* considera cada línea como una lista de campos de texto, estando dichos campos delimitados por espacios en blanco o por caracteres de tabulación. Para comparar entre si dos líneas, inicialmente se comparan por parejas todos los campos, hasta que se termina con la lista de campos, o hasta que se encuentra una diferencia. En caso de que la comparación haya llegado al final con el resultado de que ambas líneas son iguales, aún se hace una última comparación de ambas líneas carácter a carácter, tomándose el resultado final de ésta comparación. Las opciones más comunes del comando *sort* son:

-c comprueba si los ficheros de entrada están todos ordenados. Caso de no

estarlo alguno de ellos, se presenta un mensaje de error y *sort* termina con un estado de 1.

-m fusiona todos los ficheros de entrada (línea a línea) en un único fichero ordenado. Para ello es necesario que los ficheros de entrada estén ordenados. Fusionar es más rápido que ordenar, pero nótese la necesidad de que los ficheros de entrada estén ordenados.

-b ignorar los espacios en blanco al principio de cada l´ınea.

-d ignorar todos los caracteres excepto letras, números y espacios en blanco.

-f considerar las letras minúsculas como su correspondiente mayúscula

-i ignorar caracteres no ASCII.

-n considerar que los campos que tengan formato de uno o más dígitos, opcionalmente precedidos por un signo '-' y terminados en un punto decimal y un numero de dígitos, es un campo numérico y como tal se tiene en cuenta en las comparaciones.

-r ordenar en orden inverso (de mayor a menor)

-o f generar como salida un fichero con nombre *f*.

-t s considerar que los campos están delimitados por el carácter *s*

+p$_1$ **-p**$_2$ Especifica p_1 como el índice del primer campo que se usa como clave de ordenación, siendo op- cionalmente p_2 el índice del primer campo que no interviene como clave de ordenación. En caso de no especificarse p_2, se usa como clave de ordenación el resto de los campos hasta el final de la línea.

A. Comandos Linux/UNIX de manipulación de archivos y directorios

Comando	Descripción	Ejemplos
cat f_1 [. . . f_n]	Concatena y muestra los archivos	cat /etc/passwd
cd [dir]	Cambia de directorio	cd /tmp
chmod permisos fich	Cambia los permisos de un archivo	chmod +x miscript
chown usuario:grupo fich	Cambia el dueño un archivo	chown nobody miscript
cp f_1. . . f_n dir	Copia archivos	cp foo foo.backup
diff [-e] f_1 f_2	Encuentra diferencia entre archivos	diff foo.c newfoo.c
du [-sabr] f_1 [. . . f_n]	Devuelve el tamaño del directorio	du -s /home/
file f	Muestra el tipo de un archivo	file a.out
find dir test acción	Encuentra archivos.	find . -name ".bak" -print
grep expr f_1 [. . . f_n]	Busca patrones en archivos	grep druiz /etc/passwd
head -n f	Muestra las *n* primeras líneas de un archivo	head prog1.c
mkdir dir	Crea un directorio.	mkdir temp
mv f_1 . . . f_n dir	Mueve un archivo(s) a un directorio	mv a.out prog1
mv f_1 f_2	Renombra un archivo.	mv *.c prog dir
less / more fich(s)	Visualiza página a página un archivo. (less acepta comandos vi)	more /less muy largo.c
ln [-s] f acceso	Crea un acceso directo a un archivo	ln -s /users/mike/.profile .
ls	Lista el contenido del directorio	ls -l /usr/bin
pwd	Muestra la ruta del directorio actual	pwd
rm f	Borra un fichero.	rm foo.c
rm -r dir	Borra un todo un directorio	rm -rf prog dir
rmdir dir	Borra un directorio vacío	rmdir prog dir
tail -n fich	Muestra las *n* últimas líneas de un archivo	tail prog1.c
vi fich	Edita un archivo.	vi .profile

B. Comandos Linux/UNIX más frecuentes

Comando	Descripción	Ejemplos
at [-lr] hora [fecha]	Ejecuta un comando más tarde	at 6pm Friday miscript
cal [[mes] ano]	Muestra un calendario del mes/año	cal 1 2025
date [mmddhhmm] [+form]	Muestra la hora y la fecha	date
echo string	Escribe mensaje en la salida estándar	echo "Hola mundo"
finger usuario	Muestra información general sobre un usuario en la red	finger druiz@pc11.lsi.us.es
id	Número id de un usuario	id usuario
kill [-senal] PID	Enviar una señal a un proceso (dependiendo de la señal, a veces lo finalizará)	kill 1234
man comando	Ayuda del comando especificado	man gcc
passwd	Cambia la contraseña.	passwd
ps [axiu]	Muestra información sobre los procesos que se están ejecutando en el sistema	ps -ux, ps -ef
who / rwho	Muestra información de los usuarios conectados al sistema	who

C. Equivalencia de comandos Linux/UNIX y DOS

Linux	DOS	Significado
cat	type	Ver contenido de un archivo.
cd, chdir	cd, chdir	Cambio el directorio en curso.
chmod	attrib	Cambia los atributos.
clear	cls	Borra la pantalla.
ls	dir	Ver contenido de directorio.
mkdir	md, mkdir	Creación de subdirectorio.
more	more	Muestra un archivo pantalla por pantalla.
mv	move	Mover un archivo o directorio.
rmdir	rd, rmdir	Eliminación de subdirectorio.
rm -r	deltree	Eliminación de subdirectorio y todo su contenido.