Prácticas de Sistemas Operativos

Toñi Reina, David Ruiz, Juan Antonio Álvarez,
Antonio Tallón, Javier Gutiérrez, Pablo Neira, Paco Silveira,
José Ángel Bernal y Sergio Segura

Boletín ♯1: Introducción al UNIX

Curso 2008/09

Índice

---

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 "línea de comandos", 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

Cuando se quiere que el comando sea de varias líneas, se separa cada línea con el carácter barra invertida (\). Además, cuando se quiere ejecutar varios comandos en la misma línea, los separa con punto y coma (;). Por 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 boletin1.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 curso_linux.t[aeiou]x lista curso_linux.tex seleccionando la e del conjunto.

2.2. Variables de entorno

Una variable de entorno es un nombre asociado a una cadena de caracteres. Dependiendo de la variable, su utilidad puede ser distinta. Algunas son útiles para no tener que escribir muchas opciones al ejecutar un programa, otras las utiliza el propio shell (PATH, PS1,...). La tabla 2 muestra la lista de variables más usuales.

Variable	Descripción
DISPLAY	Dirección IP a donde se envían los gráficos de los clientes X.
HOME	Directorio personal.
HOSTNAME	Nombre de la máquina.
MAIL	Archivo de correo.
PATH	Lista de directorios donde buscar los programas.
PS1	Prompt.
SHELL	Intérprete de comandos por defecto.
TERM	Tipo de terminal.
USER	Nombre del usuario.

Figura 2: Variables de entorno más usuales

La forma de definir una variable de entorno cambia con el intérprete de comandos, se muestra tcsh y bash siendo los dos más populares en el ámbito Linux:

```
bash: export VARIABLE=Valor
tcsh: setenv VARIABLE Valor
```

Por ejemplo, para definir el valor de la variable DISPLAY sería:

```
bash: export DISPLAY=localhost:0.0
tcsh: setenv DISPLAYlocalhost:0.0
```

2.3. 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. Por ejemplo, si se desea realizar una transferencia de archivos por ftp automática utilizando el programa ncftp con unas determinadas instrucciones preestablecidas. Se puede crear un archivo de entrada con dichas instrucciones y ejecutar el programa de la siguiente forma:

```
$ cat > getxwpe
 open
 ftp.rediris.es
 user anonymous abc@cd.es
 cd  /sites/ftp.redhat.com/pub/redhat/linux/7.1/en/powertools/i386/RedHat/RPMS
  mget xwpe*
  bye
 ^d
$ ftp -ni < getxwpe
```

Si por ejemplo se quisiera saber los archivos que empiezan por *i* o *I* y almacenarlo en un archivo el comando ls [il]* > 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 ">>".

2.4. 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

Algunos filtros han llegado a ser tan complejos que son en si, un lenguaje de procesamiento de texto, de búsqueda de patrones, de construcción de *scripts*, y muchas otras posibilidades. Entre ellos podemos mencionar herramientas tradicionales en Linux/UNIX como *awk* y *sed* y otras más modernas como *Perl*.

2.5. Programación shell

La programación del shell es una de las herramientas más apreciadas por todos los administradores y muchos usuarios de Linux/UNIX ya que permite automatizar tareas complejas, comandos repetitivos y ejecutarlos con una simple llamada o hacerlo automáticamente a horas escogidas sin intervención de personas.

La programación shell en UNIX/Linux es, en cierto sentido, equivalente a crear archivos .BAT en DOS. La diferencia es que en UNIX/Linux es mucho más potente. Estos scripts pueden usar un sinnúmero de herramientas como:

- Comandos del sistema Linux/UNIX (ej: ls, cut)
- Funciones intrínsecas del shell (ej: kill, nice)
- Lenguaje de programación del shell (ej: if/then/else/fi)
- Programas y/o lenguajes de procesamiento en línea. (ej: awk, sed, Perl)
- Programas propios del usuario escritos en cualquier lenguaje.

El lenguaje de programación de cada shell provee de una amplia gama de estructuras de control que no serán vistas en este tema de introducción.

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[1] los demás (Otros)

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

Por ejemplo, este comando:

[1]El término "Todos"se refiere colectivamente a todas estas configuraciones: Usuario, Grupo y Otros.

```
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 líneas 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 líneas en blanco consecutivas en una sola línea 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 mostrada se produce un salto de línea. 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 línea 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 comprendido 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 línea solamente.
$	Pospuesta a cualquier expresión regular, indica que la ex-presión debe aparecer al final de la línea 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 línea.

-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 opcionalmente 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.

4. Ejercicios

1. Cree en el directorio de trabajo los siguientes directorios: bin, src y tmp.

2. Realice un comando que escriba en el archivo tmp/ficheros el listado de todos los archivos (incluidos los ocultos) que cuelguen bajo el directorio /export/home/alumnos/inf05 del sistema. Una vez generado el archivo tmp/ficheros quítele el permiso de escritura para todo el mundo (incluido el propietario). Note que tmp es el directorio que creó en el ejercicio anterior[2].

3. Copie el archivo profile-ejercicio3 que el profesor situará en el directorio /tmp del servidor de prácticas murillo a su directorio *home* con el nombre .profile. Por último, copielo a un disco flexible con el nombre profile-clase.

4. Genere automáticamente un archivo que se llame profile.num cuyo contenido sea el resultado de quitar las líneas en blanco al archivo .profile que acaba de copiar en el ejercicio anterior y enumerar las restantes. Utilice el comando more para obtener las líneas en las que aparece el patrón PATH.

5. Utilice el comando grep para realizar los siguientes filtros:

[2]Si realiza este ejercicio en el servidor de prácticas de la Escuela, conocido como murillo, tenga cuidado con el tamaño del archivo resultado

a) Directorios que contiene el directorio /usr/bin

b) Ficheros con permiso de escritura en dicho directorio.

c) Comandos del directorio /bin que contiene dos vocales seguidas en su nombre.

d) Archivos o directorios del directorio /etc que contienen, al menos, un dígito.

6. Realice un comando que muestre por la salida estándar todos los usuarios que hay conectados en el servidor de prácticas (murillo.lsi.us.es) ordenado de forma decreciente (de la ’z’ a la ’a’).

7. Realice un comando que muestre por la salida estándar todos los usuarios que hay conectados en el servidor de prácticas cuyo *login* comience por el carácter ’i’.

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] año]	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 [-señal] 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.

D. Ejercicios de Examen

1. ***(Ejercicio de Examen 1ª CONV ITI 2002-03)***

   Cuando se solicita un archivo a un servidor web (por ejemplo, Apache), se usurpa la identidad del usuario nobody, perteniciente al grupo web. Sea `personal_page.html`, una página web que contiene código PHP o ASP (código ejecutable), indique los permisos. Si usted es el usuario `i5251`, escriba para las siguientes salidas del comando `ls -l personal_page.html` el comando para ponerle los permisos más restrictivos para que pueda ser accedido correctamente por cualquiera que se conecte al servidor web.

   NOTAS:

   - No se pueden modificar los datos mostrados con el comando `ls -l personal_page.html`.
   - Solamente se pueden modificar los permisos que aparecen indicados con `?`.

   a)
   ```
   murillo:/export/home/prof/lensis/mayte> ls -l personal_page.html
   -rwx?????? 1 i5251 alumnos  256 Feb 14 9:00 personal_page.html
   ```
   b)
   ```
   murillo:/export/home/prof/lensis/mayte> ls -l personal_page.html
   -rwx?????? 1 i5251 web  256 Feb 14 9:00 personal_page.html
   ```

2. ***(Ejercicio de Examen 1ª CONV ITI 2002-03)***

   En Unix se almacena la información de los usuarios registrados en el sistema en el archivo `/etc/passwd`, cuyo formato es el siguiente:

   ```
   #nombre:contraseña:UID:GID:comentarios:directorio_home:shell_defecto
   jperez:X:1130:103:Juan Pérez:/home/jperez:/bin/bash
   ```

```
murillo:/export/home/cursos/so> cat /etc/passwd
rovayo:x:224:223:Manuel Rovayo Garcia:/export/home/prof/lensis/rovayo:/bin/ksh
druiz:x:4174:223:David Ruiz Cortes:/export/home/prof/lensis/druiz:/bin/ksh
jperez:x:243:223:Jose Antonio Perez Castellanos:/export/home/prof/lensis/jperez:/bin/ksh
reinaqu:x:4961:223:Antonia Maria Reina Quintero:/export/home/prof/lensis/reinaqu:/bin/ksh
mayte:x:5526:223:MARIA TERESA GOMEZ LOPEZ:/export/home/prof/lensis/mayte:/bin/ksh
murie:x:968:208:MURIEL CORDERO MANUEL:/export/home/alumnos/inf92/murie:/bin/ksh
orteg:x:971:208:ORTEGA AVILA DANIEL:/export/home/alumnos/inf92/orteg:/bin/ksh
oyola:x:973:208:OYOLA SANCHEZ FRANCISCO JOSE:/export/home/alumnos/inf92/oyola:/bin/ksh
pradof:x:980:208:PRADO FERNANDEZ MARCOS:/export/home/alumnos/inf92/pradof:/bin/ksh
i5562:x:2205:208:GARCIA MARIN ANTONIO:/export/home/alumnos/inf97/i5562:/bin/ksh
i5564:x:2207:208:LARA PALMA DANIEL:/export/home/alumnos/inf97/i5564:/bin/ksh
i5567:x:2208:208:POZO HIDALGO SERGIO:/export/home/alumnos/inf97/i5567:/bin/tcsh
i5576:x:2211:208:MALDONADO LOPEZ PABLO:/export/home/alumnos/inf97/i5576:/bin/bash
isabel:x:2213:208:PEREZ PEREZ ISABEL:/export/home/alumnos/inf97/isabel:/bin/bash
```

Figura 5: Extracto del archivo /etc/passwd de murillo

NOTAS:

Además de los comandos vistos en clase, le puede ser de utilidad el comando `wc`. Éste cuenta el número de líneas, palabras o letras de un archivo, y su sintaxis la siguiente:

```
wc [opción...] [archivo...]
```

Si se omite el argumento archivo, `wc` tomará los datos (naturalmente) de la entrada estándar. La lista de opciones mas importantes es la siguiente:

-c Cuenta el número de bytes.

-l Cuenta el número de líneas.

-w Cuenta el número de palabras.

Como ejemplo, se pueden contar las líneas del archivo `/etc/passwd` y de esta manera se sabrá rápidamente cuantos usuarios tiene definidos el sistema:

```
murillo:/export/home/prof/lensis/reinaqu> wc -l /etc/passwd
     2602 /etc/passwd
```

Escriba comandos Unix para responder a las siguientes cuestiones:

a) A partir del archivo `/etc/passwd` obtenga aquellos usuarios que no trabajen por defecto con el shell `/bin/ksh`.

b) A partir del mismo archivo obtenga el número de usuarios que son profesores. Note que los profesores son aquellos usuarios que tienen su directorio *home* colgando del directorio `prof`.

c) Obtenga aquellos usuarios, ordenados de forma descendente, cuyo *username* comienza por `i` y NO va seguido de cuatro números. Por ejemplo, el usuario `i5678` no debería aparecer listado, y, sin embargo, `isabel`, sí debería aparecer.

3. ***(Ejercicio de Examen 2ª CONV ITI 2002-03)***

Se pretende obtener todos los ficheros del directorio actual ordenados alfabéticamente de mayor a menor. De forma que, el resultado de realizar la operación, sería algo parecido a lo que se muestra en la Figura 6.

```
.trash
.profile
.news_time
.ddd
.bash_history
.Xauthority
.TTauthority
..
.
```

Figura 6: Resultado de la ejecución del comando

Se pide resolver el problema como un solo comando.

4. ***(Ejercicio de Examen 1ª CONV II 2002-03)***

Dada la estructura de directorios representada en la Figura 7, y suponiendo que usted se encuentra en el directorio `boletines`, escriba en una línea un comando para realizar las siguientes operaciones:

```
/
  +---home
  |    + --- practica
  |          + ---boletines
  |               + ---05-sena les
  |               |    + ---ej1
  |               |    + ---ej2
  |               |    + ---ej3
  |               + ---07-mensajes
  |               |    + ---ej1
  |               |    + ---ej2
  |               |    + ---ej3
  |               + ---08-sockets
  +---tmp
```

Figura 7: Estrcutura de directorios

a) Borre el directorio `05-senales`.

b) Copie, sin cambiar de directorio, el directorio `07-mensajes` y todo su contenido al directorio `/tmp`.

c) Póngale los permisos de acceso más restrictivos al directorio que acaba de copiar en el apartado anterior, de forma que solamente puedan hacer una copia del mismo los usuarios que pertenecen al mismo grupo de trabajo que usted.

d) Renombre el directorio `08-sockets` como `08-sockets-inet`.

e) Cree un archivo `conten.ndx` que contenga una lista ordenada alfabéticamente en orden creciente y con las líneas numeradas de todas las entradas que contiene el directorio actual.

```
i6437     pts/21       81.red  -80-33-17. Sat Jun  7 16:47  - 16:48  (00:00)
i6437     pts/19       81.red  -80-33-17. Sat Jun  7 16:45  - 16:47  (00:02)
i6437     pts/21       81.red  -80-33-17. Sat Jun  7 16:23  - 16:47  (00:23)
so        pts/20         62-36-58-151.dia Sat Jun  7 16:11   still logged in
so        pts/20       62  -36-58-151.dia Sat Jun  7 15:59  - 16:08  (00:09)
i7569     pts/19       cliente  -21721601 Sat Jun  7 14:12  - 16:28  (02:15)
i5531     pts/19       cliente  -21322702 Sat Jun  7 13:13  - 13:37  (00:24)
i5531     pts/19       cliente  -21322702 Sat Jun  7 13:11  - 13:13  (00:01)
i7698     pts/22       152.red  -81-40-19 Sat Jun  7 12:27   still logged in
```

Figura 8: Extracto de la salida del comando last

5. ***(Ejercicio de Examen 1ª CONV II 2002-03)***

   En la Figura 11 se muestra un extracto de la salida del comando `last`. A partir de la información que ofrece éste, escriba comandos para realizar las siguientes tareas:

   a) Obtenga los usuarios que se encuentran actualmente conectados, ordenados por nombre de usuario.
   NOTA:
   Los usuarios conectados son aquellos en los que aparece la cadena `still logged in`.

   b) Obtenga el número de veces que se ha conectado el usuario que vaya a ejecutar el comando.
   NOTAS:

   - Le puede ser de utilidad la variable de entorno `USER`.
   - Además de los comandos vistos en clase, le puede ser de utilidad el comando `wc`. Éste cuenta el número de líneas, palabras o letras de un archivo, y su sintaxis la siguiente:
     `wc [opción...] [archivo...]`

     Si se omite el argumento archivo, `wc` tomará los datos (naturalmente) de la entrada estándar. La lista de opciones mas importantes es la siguiente:

     -c Cuenta el número de bytes.

     -l Cuenta el número de líneas.

     -w Cuenta el número de palabras.

     Como ejemplo, se pueden contar las líneas del archivo /etc/passwd y de esta manera se sabrá rápidamente cuantos usuarios tiene definidos el sistema:

     ```
     murillo:/export/home/prof/lensis/reinaqu> wc -l /etc/passwd
         2602 /etc/passwd
     ```

6. ***(Ejercicio de Examen 2ª CONV II 2002-03)***

   En los sistemas Unix existe un archivo llamado `hosts` situado en el directorio `/etc` que contiene una asociación entre direcciones y nombres de máquinas. Un ejemplo del formato de este archivo se muestra en la Figura 9.

   Se pide escribir un comando que devuelva el número de entradas que hay pertenecientes a la subred `150.214.141`.

```
127.0.0.1       localhost
150.214.142.14  murillo         loghost
10.1.12.14      murillo -int
150.214.141.104 antena          mailhost
150.214.142.17  aleixandre      cache
150.214.141.131 casiopea        www     ftp
150.214.142.20  machado
150.214.142.21  cernuda
# Aula de Ordenadores 1
10.1.12.75      pc -12-75
10.1.12.76      pc -12-76
```

Figura 9: Extracto del archivo /etc/hosts

7. ***(Ejercicio de Examen 3ª CONV ITI 2002-03)***

   `El webmaster` de un sitio web se ha dado cuenta de que su sitio tarda mucho tiempo en cargarse en un navegador. Para aligerar este proceso, ha decidido convertir los archivos de imágenes del sitio, que estaban en formato JPEG, a formato GIF para que su carga sea más liviana. Además de modificar todos los archivos de imágenes, tendrá que cambiar todos los enlaces respectivos de las páginas web. Es decir, tendrá que modificar todos aquellos archivos en los que aparezca un enlace a un archivo .jpg.

   a) Ayude a este administrador a saber en cuántas líneas tendrá que modificar, escribiendo un comando que cuente el número de líneas en las que aparece al menos una referencia a un archivo .jpg. Todos los archivos html se encuentran en la ruta almacenada en la variable de entorno `MYWEBSITE`.

   b) Escriba un comando para ponerle los permisos menos restrictivos al **directorio que contiene todos los archivos del sitio web** para que ningún usuario, excepto el propietario, pueda acceder a estos archivos mientras que el webmaster está modificando los enlaces a las imágenes.

   c) En el directorio img que cuelga de `MYWEBSITE` están todas las imágenes del sitio web. Como ahora los archivos .jpg no van a formar parte del sitio web, se pide escribir un comando para crear una copia (de seguridad) del directorio img antes de modificar los archivos que contiene.

   NOTAS:

   - Además de los comandos vistos en clase, le puede ser de utilidad el comando `wc`. Éste cuenta el número de líneas, palabras o letras de un archivo, y su sintaxis la siguiente:

     ```
     wc [opción...] [archivo...]
     ```

     Si se omite el argumento archivo, `wc` tomará los datos (naturalmente) de la entrada estándar. La lista de opciones mas importantes es la siguiente:

     -c Cuenta el número de bytes.

     -l Cuenta el número de líneas.

     -w Cuenta el número de palabras.

8. ***(Ejercicio de Examen 3ª CONV II 2002-03)***

   Resuelva con una línea de comandos UNIX los siguientes apartados:

   a) Un usuario pretende obtener el número de ficheros que tiene en su cuenta, a los cuales sólo tiene acceso el propietario. Escriba un comando que cuente el número de archivos que cumplen esta propiedad.

   b) Escribir un comando que guarde en un fichero llamado `conexiones.txt`, la lista numerada y ordenada por orden alfabético, de los últimos 20 usuarios que se han conectado a la máquina desde la que se ejecute el comando.

   NOTAS:

   - Puede serle de ayuda el comando `last` visto en clase.

9. ***(Ejercicio de Examen 1ª CONV ITI 2003-04)***

   Antiguamente, en un servidor de prácticas de una escuela técnica, los nombres de usuarios se asignaban mediante identificadores (Ej.: `i5504`) y no como hoy en día en que se asignan según los nombres y/o apellidos de los alumnos. Además, el directorio home de un alumno se creaba dentro de otro directorio que tenía nombre indicativo del año de apertura de la cuenta (Ej.: `inf96`). Por último, si las cuentas de los alumnos tenían un tiempo de inactividad, éstas se desactivaban. Cuando el alumno volvía a renovarla, su directorio home, se colocaba en un directorio distinto al original, y relativo a la fecha de renovación de la cuenta. Por ejemplo, el alumno con identificador de usuario `i5504`, comenzó sus estudios en el año 96, fecha en la que solicitó la apertura de su cuenta. Su directorio home, inicialmente estaba situado bajo el directorio inf96. Tras un periodo de inactividad, tuvo que renovar su cuenta en el año 99, por lo que en ese momento, su directorio home pasó a encontrarse en inf99.

   NOTAS:

   - En la Figura 10 se muestra un extracto del archivo `/etc/passwd` , que muestra los usuarios del sistema.

- Suponemos que los usuarios que inicialmente comenzaron en la promoción del 98 son aquellos cuyo nombre de cuenta comienza por i6xxx (donde x es un dígito), de modo que, si miramos la Figura 10, solo el `i6217`, el `i6703` y el `i6785` (aunque este último habría renovado su cuenta en el 2000), pertenecerían a la promoción, los demás, habrían renovado su cuenta en ese año.

```
i6217:x:3800:208:CABRA FERNANDEZ MANUEL:/export/home/alumnos/inf98/i6217:/bin/ksh
i1660:x:3837:208:PALACIOS RIVERA ANTONIO:/export/home/alumnos/inf98/i1660:/bin/ksh
i4764:x:3840:208:PERALTA TENA JOAQUIN:/export/home/alumnos/inf98/i4764:/bin/ksh
i4842:x:3845:208:BARRAGAN DOMINGUEZ JUAN:/export/home/alumnos/inf98/i4842:/bin/ksh
i0297:x:3846:208:RAMIREZ HURTADO ANTONIO:/export/home/alumnos/inf98/i0297:/bin/ksh
i6703:x:3847:208:SANCHEZ ARAGON JOSE MARIA:/export/home/alumnos/inf98/i6703:/bin/ksh
i5929:x:3849:208:RUIZ HERNANDEZ RAUL:/export/home/alumnos/inf98/i5929:/bin/ksh
i1807:x:3853:208:BAENA ROCA JOSE:/export/home/alumnos/inf98/i1807:/bin/ksh
i4743:x:3855:208:MUÑOZ VALLES DANIEL:/export/home/alumnos/inf98/i4743:/bin/ksh
i5861:x:3857:208:ROLDAN SIMARRO SANTIAGO:/export/home/alumnos/inf98/i5861:/bin/ksh
i5032:x:3858:208:RAMOS RAMOS JOAQUIN:/export/home/alumnos/inf98/i5032:/bin/ksh
i6785:x:4967:208:ORTIZ SALAS JONATAN:/export/home/alumnos/inf00/i6785:/bin/ksh
```

Figura 10: Extracto del archivo /etc/passwd

Ayude a los administradores del servidor de prácticas, escribiendo un comando para cada una de las siguientes necesidades:

a) Se desea conocer aquellos usuarios que comenzasen en la promoción del 98 y hayan tenido que renovar sus cuentas.

b) ¿Cómo crearía un archivo llamado promocion98activos que contenga el comando anterior sin utilizar ningún editor de texto?.

c) Dar los permisos más restrictivos al archivo promocion98activos de manera que tanto el alumno como alguien que no pertenezca al grupo alumnos pueda ejecutar dicho archivo.

d) Usando el archivo promocion98activos, ver aquellos alumnos de la promoción del 98 activos, cuyo primer apellido sea RAMOS.

10. ***(Ejercicio de Examen 4ª CONV ITI 2002-03)***

    Ayude al administrador de un sistema Unix a escribir los siguientes comandos:

    a) Se supone definida una variable `USUARIO`. Escriba un comando para contar el número de procesos que el usuario, al que hace referencia dicha variable, tiene en ejecución.

    b) Obtener un listado de todos los archivos ocultos del directorio actual, teniendo en cuenta que en el listado NO deben aparecer las referencias al directorio actual y al directorio padre del actual. Es decir, las entradas `'.'` y `'..'`, respectivamente.

```
.trash
.profile
.news_time
.ddd
.bash_history
.Xauthority
.TTauthority
```

Figura 1

Figura 11: Ejemplo del resultado de la ejecución de este comando

11. ***(Ejercicio de Examen 2ª CONV ITI 2003-04)*** Un administrador de un sistema UNIX desea comprobar la actuación de un usuario sospechoso con cuenta en la máquina que gestiona. Dicho usuario tiene como login name el identificador `i5513`. Escriba los comandos necesarios para realizar las siguientes tareas:

a) Comprobar si está conectado en este momento.

b) Conocer cuantas veces se conectó el día 1 de agosto (No tenga en cuenta si puede haber distintos años en los que se haya conectado y fuese 1 de agosto, pero sí que solo queremos dicha fecha y no el 11 ó 21 de agosto).

c) Mostrar la información de todos los procesos que está ejecutando en este momento el usuario sospechoso.

d) Hemos visto que está ejecutando un programa que consume mucha memoria, el pid de dicho proceso es el 12362 y deseamos enviarle una señal para terminar su ejecución de manera abrupta.

NOTAS:

- El comando que muestra el historial de conexiones de los usuarios de una máquina muestra una salida con el siguiente formato:

```
i6738 pts/10    45.red-80-26-25. Sun Aug  1 17:26 - 17:31 (00:05)
i6738 pts/9     45.red-80-26-25. Sun Aug  1 17:22 - 17:31 (00:09)
i5772 pts/8     100.red-81-44-19 Sun Aug  1 17:20 - 23:21 (06:01)
carob pts/9     207.red-80-59-22 Sun Aug  1 16:30 - 16:33 (00:02)
i5523 pts/8     25.red-81-40-198 Sun Aug  1 14:18 - 17:07 (02:48)
```

12. ***(Ejercicio de Examen 2ª CONV II 2003-04)*** Se desea ayudar en la administración de un servidor web instalado en un sistema operativo UNIX. El servidor crea un archivo llamado `access_log`, situado en `/var/log/httpd` donde va almacenando un registro de las operaciones solicitadas al servidor, que normalmente consistirá en la petición de un archivo. Cada línea del archivo `access_log` contiene la siguiente información:

    a) La dirección IP de la máquina cliente que ha solicitado el servicio . (Ej: 200.69.195.9).

    b) La fecha y hora en formato GMT en la que se solicitó el servicio. (Ej: [21/Sep/2004:00:15:12 +0200])

    c) La petición realizada al servidor entre comillas (Ej.: ”GET/cursos/cursoweb/info.html HTTP/1.0”). Cada petición consta de un comando, (en este caso GET), un recurso (/cursos/cursoweb/info.html), y la versión del protocolo http con la que se trabaja (en el ejemplo, la 1.0).

    d) Un código de tres dígitos numéricos que indica el estado de la operación realizada. El código 200, indica que la operación se ha realizado con éxito. Un código 4xx o 5xx, indica que ha habido algún error en la operación.

    e) El número de bytes que se han transferido, que normalmente se corresponderán con el tamaño del archivo solicitado.

```
200.69.195.9  - - [21/Sep/2004:00:15:12 +0200] "GET /cursos/cursoweb/   info.html HTTP/1.0" 200 3085
200.69.195.9  - - [21/Sep/2004:00:15:16 +0200] "GET /cursos/cursoweb/imagenes/logo100.gif HTTP/1.0" 200 5068
150.214.141.59  - - [21/Sep/2004:10:11:53 +0200] "GET /~reinaqu/Templates/ads.js HTTP/1.1" 404 3058
200.69.195.9  - - [21/Sep/2004:00:15:19 +0200] "GET /cursos/cursoweb/estilo.css HTTP/1.0" 200 357
217.217.160.130  - - [21/Sep/2004:09:38:46 +0200] "HEAD /docencia/get.php?id=236 HTTP/1.1" 200 0
150.214.231.68  - - [21/Sep/2004:10:20:26 +0200] "OPTIONS /experto/solicitud.doc HTT    P/1.0" 200  -
80.58.11.235  - - [21/Sep/2004:10:31:58 +0200] "POST /docencia/pagina_asignatura_doc.php?id=1 HTTP/1.1" 200
19663
```

Figura 12: Extracto del archivo access_log

Se pide ayudar al administrador escribiendo comandos UNIX para realizar las siguientes operaciones:

a) Contar el número de peticiones de archivos .html que se solicitaron el 21 de Septiembre.

b) Generar un archivo llamado errores.log que contenga aquellas peticiones que hayan producido algún error. Note que las operaciones con error son aquellas con código 4xx o 5xx.

c) Muestre un listado con todos aquellas líneas que contengan peticiones distintas a GET.

d) Suponiendo definida una variable de entorno DOCHTML que contiene la ruta absoluta del directorio en el que se encuentran los archivos que exporta el servidor, copie el archivo info.html que aparece en el listado de ejemplo a su directorio home. Tenga en cuenta que la ruta que aparece en el archivo access_log es relativa al directorio que exporta el servidor.

13. ***(Ejercicio de Examen 3ª CONV II 2003-04)***

Sea la variable de entorno `USER`, suponiendo que usted es el usuario `i6482` que tiene abierta una sesión de trabajo, indique claramente cuál sería la salida de los siguientes comandos:

a) `$ echo 'Hola $USER'`

b) `$ echo "Hola $USER"`

c) Agregar a la variable `PATH` el directorio `/usr/local/bin` de modo que los comandos sean buscados en ese directorio antes que en cualquier otro

d) El comando `df -k` muestra el espacio disponible en los sistemas de archivos montados. Su salida tiene el siguiente formato:

```
Filesystem            kbytes      used    avail capacity Mounted
on
/dev/md/dsk/d2       4129290    190900  3897098     5%    /
/dev/md/dsk/d5       4129290   1129683  2958315    28%    /usr
/proc                      0         0        0     0%    /proc
fd                         0         0        0     0%    /dev/fd
mnttab                     0         0        0     0%    /etc/mnttab
/dev/md/dsk/d8       4129290   1317182  2770816    33%    /var
swap                 5559416        24  5559392     1%    /var/run
swap                 5560864      1472  5559392     1%    /tmp
/dev/md/dsk/d11     18491215   3222975 15083328    18%    /opt
/dev/dsk/c0t9d0s7   35007716   6414928 28242711    19%    /export/home
/dev/dsk/c0t10d0s7  35007716  14897664 19759975    43%    /export/home/alumnos
```

Figura 13: Resultado de la ejecución del comando df -k

Escribir un comando que muestre por la salida estándar aquellos sistemas de archivos cuyo porcentaje de ocupación esté entre el 90 y el 99 %. Tenga en cuenta que los campos están separados por espacios y que el campo `capacity` indica el porcentaje de ocupación del sistema de archivos.

14. ***(Ejercicio de Examen 3ª CONV ITI 2003-04)***

Usted es un usuario de una máquina Unix en la que tiene una cuenta. Escriba los comandos necesarios para realizar las siguientes tareas:

a) Compruebe si en el directorio raíz o en alguno de sus subdirectorios existe un archivo llamado `.history`.

b) Asigne el valor de la variable `PATH` a una variable de entorno con nombre `RUTA`.

c) Añada a la variable `RUTA` el directorio `/opt/sfw/bin` y el directorio actual.

d) Copie el archivo `profile-ejemplo` situado en el directorio `tmp` que cuelga del directorio raíz a su directorio *home*, pero con el nombre `.profile`. Utilice rutas absolutas para escribir el comando.

15. ***(Ejercicio de Examen 3ª CONV ITI 2003-04)***

A partir de la ejecución del comando `df -k` mostrada en la Figura 13*d* , escriba comandos para realizar las siguientes operaciones:

a) Obtener un listado de los sistemas de ficheros sin capacidad (`capacity = 0%`).

b) Obtener un listado de los sistemas con una capacidad superior al 40

c) Obtener un listado de los sistemas con una capacidad superior al 40

d) Suponiendo que en el directorio donde se encuentra existen 4 ficheros llamados `f01`, `f02`, `f03` y `f04`, escriba un único comando para copiar los tres primeros a una carpeta ficheros que cuelga de su directorio *home*.