

ADMINISTRACIÓN DE SERVIDORES CON LINUX

1. ESTRUCTURA DE DIRECTORIOS.

La estructura de directorios es en forma de árbol invertido.

/ --> Directorio raíz.

/usr --> Es el directorio de los usuarios. De él cuelgan, normalmente, los subdirectorios de los diferentes usuarios del sistema. Además contiene otros subdirectorios:

/usr/bin --> Proporciona algunos comandos y herramientas.

/usr/sbin --> Proporciona algunos comandos y herramientas.

/usr/man --> Contiene los manuales del sistema.

/usr/mail --> Buzón de cada usuario.

/usr/home --> Es igual al directorio de cada usuario.

/bin --> Contiene algunas órdenes de uso habitual por los usuarios del sistema.

/sbin --> Contiene algunos programas relacionados con la administración del sistema, y que por lo tanto, están reservados al superusuario.

/dev --> Contiene un conjunto de ficheros especiales destinados a la gestión de periféricos (*hda_nº* --> discos duros, *fd0* --> unidad de disco flexible, *mem* --> a toda la memoria del sistema, etc.).

/etc --> Contiene ficheros y herramientas empleados para la administración, configuración, del sistema.

/mnt --> Se emplea para montar sobre él otros sistemas de ficheros.

2. PERMISOS.

- rwx rwx rwx --> fichero con todos los permisos.

- --- --- --- --> fichero sin ningún permiso.

1 2 3 4

1 --> - fichero

d directorio

2 --> permisos del propietario.

3 --> permisos del grupo del propietario.

4 --> permisos del resto de usuarios.

- permiso desactivado.

r permiso de lectura.

w permiso de escritura.

x permiso de ejecución o **s** (*bit set uid* o *bit set gid*) con éste pueden adquirir permisos del propietario.

3. FICHEROS DE CONFIGURACIÓN.

/etc/profile Contiene la configuración de entrada de todos los usuarios.

Es común para todos los usuarios.

/usr/\$HOME/.profile Contiene la configuración que crea cada usuario para su entorno.

Este fichero, oculto, contiene un conjunto de órdenes que se ejecutan cada vez que un usuario entra al sistema, antes de mostrarle el prompt. Además de ejecutarse las órdenes contenidas en

el fichero `.profile` se ejecutan, previamente, las órdenes contenidas en el fichero `/etc/profile`, cuyo propietario es el superusuario, que es el único que puede modificar su contenido.

/etc/passwd Contiene todas las claves de cada usuario. Y tiene los siguientes campos
--> Nombre de usuario : Clave encriptada : Número de usuario : Número de grupo al que pertenece
: Un comentario : Nuestro HOME : Shell que se ejecuta cuando se arranca el sistema.

/etc/group Campos --> Nombre del grupo : Clave del grupo : Número del grupo : Todos los usuarios que forman parte de ese grupo.

/etc/resolv.conf En este fichero de configuración se debe poner:
`nameserver IP`
para establecer el servidor DNS, se aconseja poner 2 servidores DNS.

4. FICHEROS LOGS.

Los ficheros `.log` son ficheros de sucesos y se encuentran en el directorio `/var/log` o en `/root`.

Los scripts deben generar un fichero `.log` para saber lo que ha sucedido.

5. PROCESOS Y DEMONIOS.

Un **proceso** no es un programa sino un programa en ejecución, la pila del programa, las variables que cambian de valor, etc.

Un **demonio** (daemon) de un sistema multiusuario es el que se está ejecutando siempre en segundo plano, desde que se arranca el sistema hasta que se apaga, por lo que se dice que están vivos.

Algunos demonios son:

- **Cron**. Se encarga de ejecutar los programas que le indiquemos con el comando `crontab` a determinadas horas.
-- **Sendmail**. Empleado para gestionar el correo electrónico.
- **Inetd**. Es el superdemonio de Internet.

6. NIVELES DE ARRANQUE.

Una vez que se carga el Kernel en memoria, éste se encarga de ejecutar el programa `init`.

Dicho programa se encarga de inicializar los dispositivos del sistema.

Este programa se encuentra en el directorio `/etc`. Es el primer proceso, por lo que su PID es 1 y siempre está vivo, si muriese por algún motivo el sistema dejaría de funcionar.

La forma de arrancar un sistema se encuentra en el fichero `/etc/inittab`, es un fichero de texto como el `autoexec.bat`. Si queremos anular la acción `CTRL+ALT+SUPR` deberemos poner # delante de la línea *How to yor when yo press CTRL+ALT+SUPR*, del fichero `inittab`.

`init` puede arrancar en diferentes estados, algunos de ellos son:

0	Se emplea para detener el sistema.
---	------------------------------------

1	Modo de usuario individual.
2	Modo de usuario múltiple, sin procesos de servidor.
3	Modo de usuario múltiple, con procesos de servidor.
4	No utilizado.
5	Modo de usuario múltiple, con procesos de servidor y X11.
6	Reinicio del sistema.

```

/etc/rc.d/
init.d --> atd, named (etc/named.conf), nfs, sendmail (sendmail.mc),
smd (/smd.conf),
        httpd (apache, fichero de configuración: httpd.conf).
rc0.d
rc1.d
rc2.d
rc3.d
rc4.d
rc5.d
rc6.d

```

7. ÓRDENES Y COMANDOS.

Los ficheros ejecutables se encuentran en los directorios:
 /bin, /usr/bin --> los que cualquiera puede ejecutar
 /sbin, /usr/sbin --> sólo los que puede ejecutar el root
 Y se ejecutan de la siguiente forma: **./nombre_fichero**

ls muestra todos los archivos dentro de un directorio
-a ocultos
-R ver subdirectorios
-F ver tipo de fichero
-C compacto
-l muestra la capacidad en bytes y la fecha en que fueron creados
-r orden alfabético inverso
-m salida con comas
-lh tamaño en KB y en MB
-- color muestra los tipos de archivos con diferentes colores

pwd muestra la ruta donde estás

clear limpia la pantalla

man muestra un manual de ayuda del comando indicado
q pulsarla para salir de la ayuda

rm borra un fichero
-r borra un directorio no vacío
-f fuerza el borrado
-i confirma si se quiere borrar

mkdir crea un directorio

rmdir borra un directorio vacío

cd cambio de directorio
cd .. regresa al directorio anterior
cd / cambio al directorio principal de la máquina
nombre_directorio cambia a dicho directorio si está dentro del directorio en el

que te encuentras

cat muestra por pantalla el contenido de un archivo sin hace pausa
cat > fichero crea un fichero
CTRL+D para marcar el fin del fichero

more muestra el contenido de un archivo de texto por página
q si se pulsa se termina la visualización del fichero

date[option] [+format] para obtener la fecha del sistema
--set = "mes/día/año hh:mm:ss" para cambiar la fecha y la hora

df para ver las particiones del sistema, capacidad total, capacidad usada, capacidad disponible, % uso, montado.
-m en MB
-a (all) todas las particiones (/dev/hda_n° discos duros IDE y /dev/sda_n° discos duros SCASSI, /proc y /swap)

ps proporciona información acerca de los procesos que se encuentran en ejecución en el sistema en el momento en el que se ejecuta la orden. Si se ejecuta sin parámetros proporciona información únicamente sobre los procesos asociados al terminal en que se encuentra el usuario.
-x lista de todos los procesos que se están ejecutando en tu cuenta
-ax prioridad, si es un demonio, etc.
-aux w > ver.txt
-HUP PID para que ese proceso relea la configuración

pstree muestra el árbol de los procesos que dependen de otros (pid, id del proceso)

who da información sobre los usuarios que están conectados a tu estación de trabajo. Desde qué máquina se están conectando, y el tty de dicha máquina
-u da el nombre del usuario, terminal, fecha y hora, terminales que han tenido actividad hace x minutos y el PID
-b indica la hora y fecha de la última carga del sistema

crontab fichero ejecuta comandos periódicamente
-l para saber si está programado el crontab
--r para borrar la programación
-u usuario -l para saber las tareas de ese usuario
-u usuario -r para borrar las tareas de ese usuario
fichero --> #!/bin/bash
<minutos> <hora día> <día mes> <mes año> <día semana>

fichero_ejecutable
El demonio **cron** se despierta cada minuto y mira si tiene que hacer algo, sino se duerme.
Los siguientes directorios contienen las tareas:
/etc/cron.daily/ --> tareas diarias
/etc/cron.week --> tareas semanales
/etc/cron.month --> tareas mensuales
Ejemplo: fichero --> 0, 5, 10, 15, 20, ..., 55 * * *
* /root/procesos --> siempre (* * * *)
se ejecuta, cada 5 minutos el fichero procesos

top evolución que siguen los procesos durante su ejecución, cantidad de swap y RAM usada y disponible, pid, etc.
top > procesos.txt
top -b -n2>procesos.txt

ifconfig nos indica todas las interfaces de red activas (/dev/eth0, /dev/eth1 ...)

ifconfig eth0 muestra: IP, MAC, IRQ, dir E/S, nº colisiones, modo, etc.

ifconfig eth0 IP para configurar la tarjeta con esa IP

ifconfig eth0 IP broadcast IPbroadcast network IPRed netmask Máscara_Red

ifconfig eth0:0 (eth0:1, eth0:2, ...) para añadirle otra IP a la tarjeta

ifconfig eth0 down para desactivar la tarjeta

ifconfig eth0 up para activarla

ifconfig eth0 hw ether MAC para cambiarle la MAC a la tarjeta

ifconfig eth0 promisc para ponerla en modo promiscuo

ifconfig eth0 -promisc para desactivar el modo promiscuo

El fichero **/etc/network/interfaces** contiene toda la información de las interfaces de red,

```

ejemplo del contenido de dicho fichero -->
                                     iface eth0 inet static
                                         address IP
                                         netmask IP1
                                         network IP2
                                         broadcast IP3
                                         gateway IP4
                                     iface eth1 inet

```

static ...

ifup eth0 para activar la tarjeta

ifdown eth0 para desactivar la tarjeta

route muestra la tabla de enrutamiento

-n aparece la IP en la tabla actual de enrutamiento

add -host IP eht0[:0] [:1] ...

del -host IP eht0[:0] [:1] ...

add -net w.x.y.0 (IPRed)

add default gw IP

Ejemplos de configuración de un router:



```

eth0 192.168.9.0 --> ifconfig eth0 192.168.9.0
eth1 192.168.7.0 --> ifconfig eth1 192.168.7.0
eth2 192.168.14.0 --> ifconfig eth2 192.168.14.0

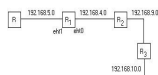
```

```

route add -net 192.168.9.0 netmask 255.255.255.0 gw 192.168.9.1 eth0
route add -net 192.168.7.0 netmask 255.255.255.0 gw 192.168.7.1 eth1
route add -net 192.168.14.0 netmask 255.255.255.0 gw 192.168.14.1 eth2
route default gw 192.168.14.10 metric 1 --> nº de saltos para llegar a

```

ese router



route add -net 192.168.5.0 eth1 --> añade la red 5.0 que pertenece a esa tarjeta

route add -net 192.168.4.0 eth0 --> añade la red 4.0 con la primera interfaz, eth0

route add -net 192.168.9.0 gw 192.168.4.2 --> añade una dirección IP a un host que pertenece al gw indicado

route add -net 192.168.10.0 gw 192.168.4.2

netstat para saber los puertos que hay abiertos

-eop programas que utilizan esos puertos, entre otros datos
-c información continuada, se va refrescando
-ln
-a todos los abiertos, conectados o no
-i aquellos que están escuchando
-- route para ver la tabla de enrutamiento del sistema
-- tcp
-- udp

ping IP/URL permite saber el estado de una máquina. Entre otros datos obtenemos el tiempo que tarda un paquete en llegar a su destino

traceroute IP/URL

last para saber los últimos usuarios que han entrado por consola y por ftp

lastb para saber los usuarios que han intentado entrar y no han podido

find permite localizar ficheros según un determinado criterio, para realizar a continuación alguna acción

find nombre_directorio criterio_búsqueda acción

criterios --> **-name fichero** busca el fichero indicado
-user usuario busca aquellos ficheros que pertenecen

al usuario

-group grupo busca los ficheros que pertenecen al

grupo de

usuarios indicado

-perm [-] m localiza los ficheros cuya máscara sea

exactamente

m. Con - busca los que tengan como

mínimo los derechos

indicados en la máscara.

-mtime n localiza los ficheros modificados hace n

días

-mtime -n localiza los ficheros modificados en los

últimos n días

-mtime +n los ficheros modificados hace más de n

días

-size m, -m, +m busca aquellos ficheros que ocupan en menos de m o más de m bloques. Si m va entonces se buscan los que tengan ese

disco m,

seguido de c

tamaño en bytes.

-type x busca los ficheros de dispositivo de bloque

(x=b), de

caracteres (x=c), directorios (x=d), pipes

(x=p), ligaduras

simbólicas (x=l) o ficheros reguales (x=f)

acciones --> **-print** muestra en pantalla los nombres de los ficheros localizados

indicado

-exec programa \; ejecuta el programa

whereis nos indica la localización exacta de un fichero o de una orden o comando

grep patrón fichero/s permite localizar una palabra clave o una frase en varios ficheros,

indicando las líneas y ficheros en los que aparece

-i ignora la distinción entre mayúsculas y minúsculas

-n cada línea visualizada se acompaña del n° de línea

correspondiente en el fichero

- v** indica que se muestren las líneas que no encajen en el patrón
- l** indica que se muestren los nombres de los ficheros donde se ha

encontrado en

patrón

ps aux|grep syslog à para comprobar si corre el demonio syslog

diff archivo1 archivo2 compara los archivos señalados línea a línea

nice para cambiar la prioridad a un proceso

-n 19 ./comando para darle la menor prioridad

-n 20 ./comando para darle la mayor prioridad

renice cambia la prioridad de un proceso en ejecución

tar -czvf [arch.tgz] [directorio] permite comprimir los archivos de un directorio

-xvf descomprime un archivo con extensión .tar

gunzip permite descomprimir un archivo con extensión .tgz

mail para visualizar el correo almacenado en el buzón. Pasos para enviar un mensaje: 1º mail nombre_usuario

mensaje

CTRL+D

chmod [-R] abc archivo cambia el modo de uso (protección) de un archivo, donde *a* indica

permisos para el propietario, *b* permisos para el grupo y *c* permisos para otros usuarios. Dichas letras tomaran los valores sumando los permisos:

0 --> ningún permiso

1 --> permiso de ejecución (x)

2 --> permiso de escritura (w)

4 --> permiso de lectura (r)

Ejemplo: **chmod 777 archivo.txt** --> otorga todos los permisos a cualquier usuario

chown [-R] propietario archivo cambia el propietario asignado a archivo, o directorio (-R).

chgrp [-R] archivo cambia el grupo asignado a archivo

umask máscara para modificar la máscara de derechos por defecto
máscara --> valor derechos (0664) - 0666 = máscara (0664)

su permite crear un shell con derechos efectivos correspondientes a otro usuario, del cual conocemos el password

login para iniciar una sesión

logout para terminar una sesión

halt para apagar el sistema

shutdown para apagar el sistema si eres el root

-r para reiniciar el sistema

-h para apagarlo

reboot para reiniciar el sistema

kill -señal PID para matar procesos

Señal	Descripcion
-1	aborta los procesos
-9	permite matar un proceso especificado por un nº de proceso (PID)
-15	envía una señal a un proceso para que termine, pero si está preparado puede ignorar la señal, opción por defecto
-HUP	para que ese proceso relea la configuración

adduser para añadir un usuario
-- home ruta (Ejemplo de ruta: /home/usuarios/) para indicar donde se guardarán los ficheros del usuario
-- ingroup nombre--_grupo nombre_fichero se indica el grupo en el que se guardará, si no existe el fichero, lo creará

addgroup para añadir un grupo (/etc/group/)

deluser para borrar un usuario
-r borra el usuario y además su cola de correo (/var/spool/mail/) y su directorio home

delgroup para borrar un grupo

modconf para instalar o desinstalar módulos

tasksel para instalar grupos de paquetes

pppconfig para configurar el módem

apt-get para instalar, desinstalar o actualizar paquetes
apt-get update&&apt-get upgrade para actualizar paquetes en Debian
apt-get install para instalar un paquete
apt-get remove para borrar un paquete

touch nombre_fichero crea un fichero de texto vacío
touch /etc/nologin --> si se crea este fichero, sólo podrán entrar en el sistema los usuarios que tengan uid=0 y gid=0. Al borrarlo el sistema vuelve a dejar acceso a usuarios normales.
touch script --> el contenido del script debe ser: #/bin/bash comandos
y después se debe dar derecho de ejecución, **chmod**

init nº arranque

id para obtener datos sobre tu usuario

alias nombre='orden' para sustituir un valor por un nombre más fácil de recordar. Se almacenan en el fichero /etc/profile
Ejemplo: **alias cls='clear'**

unalias nombre_alias para eliminar un alias creado

passwd permite cambiar tu password

lock bloquea el terminal, para ello pide un password, dos veces

test mediante esta orden podemos evaluar expresiones condicionales. Su

función es evaluar una expresión condicional, de forma que si ésta resulta cierta devuelve un `errorlevel` igual a 0, y si es falsa devuelve un `errorlevel` distinto de 0.

-r fichero	Evalúa a cierto si el fichero existe y tiene permiso de lectura para el usuario que accede.
-w fichero	Evalúa a cierto si el fichero existe y tiene permiso de escritura para el usuario que accede.
-x fichero	Evalúa a cierto si el fichero existe y tiene permiso de ejecución para el usuario que accede.
-f fichero	Evalúa a cierto si el fichero existe y es un fichero regular.
-d fichero	Evalúa a cierto si es un directorio.
-c fichero	Evalúa a cierto si el fichero existe y es un fichero especial de caracteres.
-b fichero	Evalúa a cierto si el fichero existe y es un fichero especial de bloques.
-h fichero	Evalúa a cierto si fichero existe y es una ligadura simbólica.
-z s1	Evalúa a cierto si la longitud de la cadena s1 es 0.
-n s1	Evalúa a cierto si la longitud de la cadena s1 es distinta de 0.
s1 = s2	Evalúa a cierto si ambas cadenas son idénticas.
s1 != s2	Evalúa a cierto si ambas cadenas no son idénticas.
s1	Evalúa a cierto si s1 no es la cadena vacía.
n1 -eq n2	Evalúa a cierto si los dos enteros (n1, n2) son exactamente iguales.
n1 -lt n2	Evalúa a cierto si el entero n1 es estrictamente menor que n2.
n1 -gt n2	Evalúa a cierto si el entero n1 es estrictamente mayor que n2.
n1 -le n2	Evalúa a cierto si el entero n1 es menor o igual que n2.
n1 -ge n2	Evalúa a cierto si el entero n1 es mayor o igual que n2.
n1 -en n2	Evalúa a cierto si el entero n1 es distinto de n2.

Además, pueden emplearse los siguiente operadores para construir expresiones

condicionales compuestas:

!	Operador "not"
-a	Operador "and"
-o	Operador "or"
()	Construcción de subexpresiones

sleep es posible utilizar esta orden para suspender temporalmente la ejecución de un *shell*, y de esa forma retrasar la ejecución de algún comando.

cp [ruta/]archivo a copiar [ruta destino/]nombre nuevo

mv mueve un archivo de un directorio a otro, o cambia el nombre a un archivo

sed nos permite procesar el contenido de un fichero de texto línea a línea,

sometiendo las mismas a determinadas acciones, enviándose el resultado de este procesamiento a la salida estándar.
 La sintaxis básica es: `sed [opciones] 'comandos-sed' fichero`

.	Representa un carácter cualquiera.
[...]	Puede ser reemplazado por cualquier carácter de los encerrados entre corchetes.
[^...]	Puede ser reemplazado por cualquier carácter no incluido en la lista.
^...	Considera únicamente aquellas líneas que comiencen por la expresión que sigue al ^.
...\$	Considerará únicamente aquellas líneas que acaben en la expresión que precede al \$.
\c	Se reemplaza por el carácter c, al margen del significado especial que los comandos grep o sed asocian a dicho carácter para construir expresiones regulares.

Formatos de sustitución.

`sed` acepta las siguientes opciones :

- `-n` Suprime las salidas al dispositivo de salida estándar, salvo que se indique de forma explícita la visualización en la acción de procesamiento.
- `-f fichero` Las acciones a realizar se obtienen desde el fichero indicado.

Para borrar la línea 2 de un fichero: `$sed '3d' fichero`
 Para borrar las líneas de un fichero que contengan una palabra:
`$sed '/palabra/d' fich`
 Además, podemos utilizar "`\("` y "`\)"` para delimitar campos y referenciarlo posteriormente mediante "`\1`".

cut se emplea para eliminar una parte en cada línea de un fichero de texto, ya sea atendiendo a una división en campos, o bien según una delimitación de columnas.

`cut [opciones] -b lista_columnas -f lista_campos`

Por defecto se emplea como delimitador un tabulador, para establecer un delimitador

distinto se emplea la opción `d`
`$cut -d: -f 1-2 fichero`

paste para unir las líneas de dos ficheros.

wc cuenta los caracteres, palabras y líneas del archivo de texto indicado.

awk es una herramienta que se emplea para procesar ficheros de texto, línea a línea, dividiendo la línea en campos, y realizando diversas acciones sobre ellos. Para ello se especifican dichas acciones en términos de un programa `awk`. En consecuencia, resulta muy útil en la implementación de shell-scripts.

La sintaxis básica es la siguiente:

`awk [-Fc] [-v var=valor...] [-f fich_programa|'programa']`

[ficheros_texto]

donde :

- Fc Asigna el carácter separador de los campos de las líneas (por defecto es un blanco).
- f *fichero* Hace que el programa que requiere awk se tome desde un *fichero* en lugar de un texto entrecomillado.
- v *var=valor* Asigna la variable *var* al valor indicado. Sin embargo, la variable *var* sólo puede ser accedida en una sentencia BEGIN del awk (que incluye las acciones que awk debe ejecutar antes de empezar a procesar el fichero o ficheros de texto de entrada).

El programa awk consta de sentencias de la forma: `patrón{acción}`

Para cada línea del fichero de entrada se hace el siguiente procesamiento: se contrasta la línea con cada patrón de los que aparecen en el programa awk. Si encaja en él se realizan las acciones que siguen a ese patrón entre las llaves. Cuando no quedan más patrones en los que encaje se pasa a la línea siguiente. El patrón puede no aparecer, en cuyo caso todas las líneas encajan en él. Si no se indica ninguna acción. Entonces se visualiza en la salida estándar la línea completa.

Para formar los patrones se emplean combinaciones lógicas (utilizando operadores lógicos como los de C, ||, &&, !) de expresiones relacionales y expresiones regulares (similares a las empleadas en el programa grep).

También pueden emplearse expresiones regulares para formar los patrones, pero en este caso deben ir encerrados entre barras hacia la derecha: `"/expr_regular/"`.

Además, awk proporciona dos patrones especiales: BEGIN y END que pueden emplearse para realizar acciones antes de que la primera línea se leída y procesada y después de que la última línea sea leída y procesada.

Awk reconoce entre otras las siguientes variables:

ARGC	Nº de argumentos (incluye el propio programa awk y los ficheros argumentos).
ARGV	Array de argumentos.
FILENAME	Nombre del fichero que está procesando actualmente.
FNR	Nº de registro actual en el fichero que está siendo procesado en el momento actual.
NF	Nº de campos en un registro.
FS	Separador de campo a la entrada.
OFS	Separador de campo a la salida.

RS	Separador de registro a la entrada.
ORS	Separador de registro a la salida.
\$0	Registro completo actual.
\$1, \$n	Campos en el registro actual.

Las acciones que aparecen encerradas entre las llaves tienen una sintaxis similar al C:

```

if (expresión) sentencia [else sentencia]
while (expresión) sentencia
for (expresión;expresión;expresión) sentencia
for (var in nombre_array) sentencia
delete nombre_array[índice]
break
continue
{[sentencia]}
variable=expresión
print [lista de expresiones] [ > fichero ]
printf formato [,lista de expresiones] [ > fichero ]
next      (pasa al siguiente registro, sin procesar el actual)
exit      (saltarse el resto de registros del fichero de entrada)
return expresión

```

Para formar las expresiones se emplearan los operadores de C. se permite el uso de

arrays, incluso multidimensionales. Además, awk soporta un conjunto de funciones

predefinidas entre las cuales están:

index(s, t) Si t es subcadena de s, devuelve la posición de comienzo de t en s. en caso contrario devuelve 0.

length(s) Devuelve la longitud de la cadena s.

substr(s, m, n) Devuelve una subcadena de s tomando n caracteres

desde la

posición m.

getline Lee el siguiente registro de entrada, y asigna \$0 a dicho registro.

system(comando) Ejecuta el comando desde un shell y retorna el valor de

errorlevel obtenido.

head -n° Obtiene las n primeras líneas de un fichero de texto.

tail -n° Obtiene las n últimas líneas de un fichero de texto.

cmp archivo1 archivo2 Compara dos archivos señalados.

diff archivo1 archivo2 Compara los archivos señalados línea a línea.

file archivo Determina el tipo (texto, programa, etc) del o de los archivos indicados.

uniq fichero Elimina las líneas duplicadas de un fichero ordenado alfabéticamente.

sort Ordena ficheros de texto alfabéticamente, presentando el resultado en la salida

estándar. Si se indican varios ficheros argumento, el programa realiza la ordenación

sobre la unión de todos ellos.

time [opciones] comando [argumentos] Devuelve el tiempo que tarda en

ejecutarse comando

finger dato Permite saber información sobre el o los usuarios que cumplan con dato.

Dato puede ser el login, apellido, nombre, etc.

-m login Te da información sobre el usuario que tenga dicho login, la última vez que se conectó, etc.

@maquina Te da información sobre todos los usuarios que estén conectados a maquina.

chfn Permite cambiar tu configuración finger.

tar -czvf Permite comprimir los archivos de un directorio.

-xvf Descomprime un archivo con extensión .tar

gunzip Permite descomprimir un archivo con extensión .tgz

mdir Muestra el contenido de un diskette.

mdel Borra los archivos de la unidad A.

mmd Crea un directorio en el disco A especificando la ruta.

mcd Permite cambiarte de directorio dentro de la unidad A.

mcopy Permite copiar archivos desde y hacia una Unidad (mcopy "a:/manual.txt" copia ese fichero en el directorio en el que te encuentres).

history Muestra los últimos n comandos que se ejecutaron en la línea de comandos.

!! Repite él último comando que se ejecutó.

! Repite el comando que tenga dicho número en el historial.

write usuario [terminal] Enviar mensajes

at hora [fecha] [+incremento] Planificar trabajos para una ejecución posterior.

-l [trabajo...] lista todos los trabajos asignados actualmente por el usuario que invoca

la orden

-r trabajo... para cancelar trabajos

fecha --> today, thursday next week (el jueves de la próxima semana),

incremento --> n° (minutes|hours|days|weeks|months|years)

batch < script Planifica trabajos para su ejecución diferida, es decir, cuando la carga del

sistema sea lo suficientemente baja.

NOTA.- Se debe utilizar la orden **at -r** para eliminar un trabajo planificado

con *batch*.

Redireccionamientos.

> Redirecciona la salida a un fichero

>> Añade información al fichero

2> Redirecciona la salida de errores, normalmente se usa comando

2> /dev/null

Pipes.

```

&      Para ejecutar varias órdenes de forma simultánea
;      Se ejecutan de forma simultánea
derecha &&      Se ejecuta la de la izquierda y si errorlevel = 0 la de la
derecha ||      Se ejecuta la de la izquierda y si errorlevel != 0 la de la
derecha
opciones -->   c      Crea un shell para el comando
                  e      Abandona el shell cuando falle una orden
                  t      Abandona cuando ejecute una orden

```

8. IMPLEMENTACIÓN DE SHELL-SCRIPTS.

Son ficheros de texto que indican el shell que se va a utilizar (#! Shell) y una serie de comandos.

```

/bin/bash
/bin/sh --> enlace simbólico del anterior shell
/bin/pol
Las formas de ejecutar un script son:  ./nombre_script
                                         sh nombre_script
                                         /root/nombre_script

```

ctrl + z --> para la ejecución

fg %n° --> devuelve el número del proceso que se ha parado y vuelve al procesador

de textos

jobs --> trabajos que se están ejecutando

bg %n° --> para lanzar un proceso en segundo plano

\$0fichero.temp --> crea un fichero temporal

a) Uso de variables. En un script podemos usar variables para guardar valores, para ello no hace falta que declaremos la variable que vamos a utilizar, como lo haríamos en un lenguaje de programación simplemente le ponemos un nombre y la usamos...

Ejemplo:

```

VARIABLE=5
echo $VARIABLE

```

Como vemos, siempre que utilicemos la variable tendremos que poner el caracter dolar (\$) delante.

variable=`comando` --> ejecuta el comando y lo guarda en esa variable

Variables del shell

\$?	error level, 0 si todo ha ido bien y otro n° indica que se ha producido un error
\$\$	PID del shell
\$_	PID del último proceso lanzado en background
\$-	opciones actuales del shell
\$#	Número de argumentos de un shell-script, es decir, número de parámetros
\$@	Argumentos de un shell-script
\$0	Nombre del comando o del script en ejecución
\$1	Primer parámetro

b) Construcción IF-THEN-ELSE. En los scripts se pueden hacer condiciones usando IF, su sintaxis

es:

```
if <condicion>                o bien    if <condicion>
  then                          then
  <comando1>                    <comando1>
  <comando2>                    <comando2>
  ...                           else
  fi                             <comando3>
                                <comando4>
                                fi
```

Ejemplo:

```
VARIABLE=0
if echo $VARIABLE
then
  echo -e "\n\tHemos entrado en THEN\n"
else
  echo -e "\n\tHemos entrado en ELSE\n"
fi
```

c) Construcción FOR. Sintaxis:

```
for nombre [in palabra palabra ...]
do
  <comando1>
  <comando2>
  ...
done
```

Ejemplo:

```
for VARIABLE in 1 2 3 4 5
do
  echo $VARIABLE
done
```

d) Construcción WHILE. Sintaxis:

```
while condicion                ejemplo:    while true
do                              do
  <comando1>                    echo "Dentro de While"
  <comando2>                    done
  ...
done
```

e) Construcción SELECT CASE. Veámoslo usando un ejemplo:

```
VALOR=start
case $VALOR in
  start)
    echo -e "\n\tHemos entrado en start"
    ;;
  stop)
    echo -e "\n\tHemos entrado en stop"
    ;;
  restart)
    echo -e "\n\tHemos entrado en restart"
    ;;
  reload)
    echo -e "\n\tHemos entrado en reload"
    ;;
```

```
*)
    echo -e "\n\tEntramos aqui cuando VALOR no contiene ningun valor
de los anteriores"
    echo -en "\n\tEl valor introducido es: " $VALOR
    ;;
esac
```

f) Pedirle un dato al Usuario. En la administración de sistemas, los scripts se suelen programar para realizar operaciones automáticamente sin interacción con el usuario. Es decir, no suelen pedir datos o hacer preguntas, si necesitan alguno, se les pasan por parámetro en la llamada del script.

No obstante, con la instrucción **read** podemos hacerlo.

Ejemplo:

```
read VARIABLE
echo "El dato introducido es: $VARIABLE"
```

g) Algunos procesos que podemos encontrar son:

httpd	Servidor Apache
sendmail	Servidor SMTP
sshd	Servidor SSH
xinetd	SuperServidor