LINUX. GESTIÓN DE USUARIOS Y GRUPOS. GESTIÓN DE PROCESOS.

1 USUARIOS Y GRUPOS

Los usuarios se identifican por un número de usuario llamado **UID** (User ID). El UID es un número entero distinto a cualquier otro UID de otro usuario, que lo identifica específicamente en el sistema.

Cada usuario debe pertenecer a un grupo obligatoriamente, llamado grupo primario o principal, y además puede pertenecer a otros grupos, llamados grupos secundarios.

Un grupo es un conjunto de usuarios. Puede haber grupos que solo tengan a un usuario. Cada grupo se identifica por el **GID** (Group ID) que es también un número entero que lo identifica y los distingue de los demás grupos.

1.1 TIPOS DE USUARIOS

Entre los usuarios dados de alta en el sistema, podemos distinguir entre tres tipos de usuarios:

- Usuario **root**. También se le denomina superusuario o administrador. Es el usuario que cuenta con todos los privilegios sobre el sistema. Tiene acceso total a todo el sistema, se encarga de la administración del mismo, su actualización y mantenimiento. Se identifica porque su UID es el 0.

- Usuarios del **sistema**. No son en el sentido físico del término. Se generan al instalar el sistema o cuando se instala cualquier servicio. Aunque no tienen todos los privilegios, tienen algunos dependiendo del usuario que se trate. No son usuarios que entran al sistema e inician una sesión, por lo que no suelen tener una Shell bash de inicio de sesión, como veremos más adelante. El UID que tienen es mayor a 1 y menor a 1000, salvo por el usuario nobody al que se le asigna el último UID posible, el 65534.

- Usuarios **normales**. Son los usuarios que se conectarán al sistema introduciendo su nombre de usuario y contraseña. Tienen un directorio de trabajo dentro del directorio /home que es donde trabajan y dentro del cual tienen todos los privilegios. El UID que se les asigna es a partir del 1000, aunque es configurable. Al instalar el SO se crea uno por defecto. Un usuario normal solo tiene acceso a los archivos de su propiedad o a los que se les haya dado permiso.

Si en el sistema solo hay un usuario normal, podemos configurar para que no pida usuario y contraseña, sino que entre directamente como ese usuario.

1.2 FICHEROS DE CONFIGURACIÓN

Son los ficheros que el sistema lee o modifica a la hora de gestionar los usuarios y grupos.

/etc/passwd

Este fichero contiene en cada una de sus líneas información sobre el usuario. La información se organiza en campos separados por el carácter ":". Cada línea del fichero tiene la siguiente estructura:

```
usbmux:x:113:46:usbmux daemon,,,:/home/usbmux:/bin/false
kernoops:x:114:65534:Kernel Oops Tracking Daemon,,,:/:/bin/false
statd:x:115:65534::/var/lib/nfs:/bin/false
sshd:x:116:65534::/var/run/sshd:/usr/sbin/nologin
rtkit:x:117:118:RealtimeKit,,,:/proc:/bin/false
saned:x:118:310::/home/saned:/bin/false
timidity:x:119:119:TiMidity++ MIDI sequencer
service:/etc/timidity:/bin/false
hplip:x:120:7:HPLIP system user,,,:/var/run/hplip:/bin/false
gdm:x:121:121:Gnome Display Manager:/var/lib/gdm:/bin/false
professor:x:1000:1000:professor,,,:/home/professor:/bin/bash
```

Donde cada campo tiene el siguiente significado:

login: nombre del usuario

x: la x significa que hay una contraseña almacenada y encriptada en el fichero /etc/shadow

UID: número entero igual o superior a 0. El 0 está reservado al usuario root. Los números del 1 al 99 están reservados para usuarios especiales del sistema. Para los usuarios normales se utiliza el rango del 1000 en adelante.

GID: número entero igual o superior a 0. El 0 está reservado al grupo de root. Este número representa el número del grupo primario al que pertenece el usuario.

información: información sobre el usuario. Se muestra separada por comas. Puede no contener ninguna información, por lo que entonces contendría una serie de comas. A este campo se le llama GECOS.

directorio_personal: ruta absoluta del directorio personal del usuario.

shell_de_inicio: ruta absoluta de la shell de entrada por defecto para el usuario. La que más se utiliza es la shell bash, aunque hay más. Si la shell es /bin/false, también llamada shell nula, el usuario podría entrar en el sistema pero no a través de una sesión, sino que podría ejecutar ciertos procesos a su nombre como mail, ftp, news, ssh...

/etc/shadow

En este fichero están las contraseñas de cada usuario encriptadas. Este fichero debe tener una especial seguridad, por lo que solo root o un usuario que tengas sus privilegios puede tener permiso de lectura sobre él.

Para las contraseñas en Linux, el usuario que la ha creado, cada vez que entre en el sistema, escribe su contraseña y el sistema comprueba su contraseña con la que tiene almacenada en el fichero shadow.

En sistemas Linux la contraseña encriptada se solía guardar en el campo **x** del fichero /etc/passwd, utilizando un algoritmo de encriptación llamado DES (Data Encryption Standard) que encripta las contraseñas utilizando la función **crypt** de Linux. Por motivos de seguridad, se empezó a utilizar el fichero /etc/shadow, y algoritmos de encriptación más complejos. A la forma de utilizar este fichero para almacenar las contraseñas se le llama **shadow password**.

Los algoritmos de encriptación más utilizados en Linux son DES, MD-5, SHA-256 y SHA-512.

Para saber cuál es, en el archivo shadow, en el campo de la contraseña, miramos los caracteres de la contraseña, y así sabremos el algoritmo utilizado:

DES son 13 caracteres
MD5 si empieza por 1
SHA-256 si empieza por 5
SHA-512 si empieza por 6

Cada línea del fichero tiene la siguiente estructura:
login:contraseña:últ_cambio:mín:máx:aviso:inactividad:exp:reservado

Donde cada campo tiene el siguiente significado:

login: nombre del usuario
contraseña: contraseña del usuario cifrada
últ_cambio: fecha última en la que se cambió la contraseña
mín: mínimo número de días antes de poder volver a cambiar la contraseña
máx: máximo número de días que la contraseña puede estar activa. Se utiliza para obligar al usuario a cambiar la contraseña.
aviso: el número de días de antelación con el que se avisará al usuario de la expiración de su contraseña
inactividad: cantidad de días de inactividad, es decir, sin entrar al sistema hasta que la cuenta expire.
exp: fecha en que expira la cuenta. Se cuenta en días a partir del 01-01-1970. Si el campo está en blanco, no expira nunca.
reservado: campo reservado para usos en el futuro.

/etc/group

Fichero donde se encuentran los grupos definidos en el sistema.
Cada usuario del sistema debe pertenecer obligatoriamente a un grupo principal o primario. El grupo principal de cada usuario es el grupo cuyo GID viene en el fichero /etc/passwd.

Además, un usuario puede pertenecer a otros grupos, llamados grupos secundarios.

Cada línea del fichero tiene la siguiente estructura: **grupo:x:GID:usuarios:**

```
root:x:0:
adm:x:4:usuario
tty:x:5:
disk:x:6:
lp:x:7:
dialout:x:20:usuario
cdrom:x:24:usuario
admin:x:121:usuario
usuario:x:1000:
```

Donde cada campo tiene el siguiente significado:

grupo: contiene el nombre del grupo
x: se utilizaba sobre todo en los sistemas Unix para almacenar la contraseña de grupo. Actualmente no se suele utilizar, salvo para proteger ciertos grupos de que un usuario que no pertenezca a él.
GID: tiene el mismo significado que en el fichero passwd. Número que identifica al grupo en el sistema.
usuarios: lista separada por comas de usuarios que tienen a ese grupo como grupo secundario. Para saber si algún usuario tiene el grupo como primario, deberíamos mirarlo en el fichero passwd.

/etc/gshadow

Fichero donde se guardan las contraseñas de los grupos del sistema. Aunque las contraseñas no se utilicen para los grupos, es necesario el fichero para proteger al grupo. Al igual que shadow, solo root tiene permiso de lectura sobre el fichero.

Cada línea del fichero tiene la siguiente estructura: **nombre:contraseña:**

nombre: nombre del grupo
contraseña: puede ser una contraseña encriptada o bien los comodines * o !, dependiendo de si queremos usar las contraseñas de grupo o no.

/etc/default/useradd

Contiene los valores por defecto a la hora de añadir un usuario al sistema con el comando useradd.

/etc/adduser.conf

Contiene los valores por defecto cuando se añaden usuarios con el comando adduser.

/etc/deluser.conf

Contiene los valores por defecto cuando se eliminan usuarios con el comando deluser.

/etc/login.defs

Si se utiliza el sistema de contraseñas encriptadas, este archivo define algunos valores por defecto sobre la encriptación de contraseñas y otros parámetros al generar un usuario nuevo.

/etc/shells

El fichero de shells contiene una lista de shells válidos. Si la shell de usuario no está en este fichero, o bien el usuario tiene la shell /bin/false, no puede acceder al sistema mediante login.

DIRECTORIOS DE CONFIGURACIÓN

/etc/skel

Directorio que almacena el contenido del directorio de los nuevos usuarios que se vayan añadiendo al sistema.

1.3 GESTIÓN DE USUARIOS Y GRUPOS EN MODO TEXTO

Editor de ficheros

Para ver el contenido de los ficheros de configuración, podemos usar los comandos vistos hasta ahora para ver el contenido de los ficheros, como cat, more, less, head o tail. Pero si además queremos cambiar algún valor de ellos, si tenemos permiso para ello, debemos usar un editor de textos.

Desde el modo gráfico, al abrir un fichero de texto, lo hace utilizando el programa gedit, pero desde la terminal, podemos, además de gedit, utilizar nano, vi, o cualquier otro que queramos instalar.

1.3.1 Comandos para cambiar de usuario o ejecutar comandos con privilegiosde administrador

Dentro de la terminal podemos cambiar de usuario mediante el comando:

su

> Nos permite entrar como el usuario que indiquemos, o root si no indicamos ninguno (en este caso debemos ejecutarla con sudo, es decir, sudo su).

sudo

Permite ejecutar comandos como si fuéramos root. Previamente el usuario deberá estar configurado en el fichero /etc/sudoers, como usuario que puede tener privilegios de root, o bien pertenecer al grupo que se especifique en ese fichero.

Argumento:
comando → es el comando que ejecutaremos con los privilegios de superusuario. Nos pedirá la clave de root antes de ejecutarlo. Durante un tiempo después de introducir la clave podremos ejecutar otros comandos con privilegios de root sin que nos la vuelva a pedir.

Ejemplo:

Cambia la contraseña de root desde tu propio usuario. Entra como root. Mira el contenido del fichero /etc/shadow y /etc/gshadow de forma paginada por pantalla. Mira el contenido de /etc/sudoers y comprueba quién tiene o puede tener todos los privilegios del sistema.

Solución:

```
sudo passwd root
su
less /etc/shadow
more /etc/gshadow
cat /etc/sudoers
```

1.3.2 Comandos para la gestión de usuarios y grupos

Vamos a ver los diferentes comandos que podemos usar para la gestión de usuarios y grupos. Hay que tener en cuenta que solo el administrador o un usuario con privilegios de administrador pueden añadir, eliminar o modificar la configuración de los usuarios y grupos, por lo que desde el terminal tendremos que entrar como root, o bien, como usuario utilizar el comando sudo.

whoami

Muestra el nombre de usuario asociado con el ID efectivo del usuario actual.

adduser

Añade usuarios al sistema. También añade un usuario existente a un grupo que también exista previamente en el sistema.
Existe también el comando **useradd**, aunque éste añade también usuarios, hay que indicarle después la contraseña, ya que añade un (!) en el fichero /etc/shadow y crearle el directorio en home o bien especificándoselo en las opciones.

Sintaxis:
adduser [opciones] usuario
adduser [opciones] usuario grupo

Opciones:
--ingroup nomgrupo → crea al usuario con nomgrupo como grupo principal.
--gid num_gid → igual que la opción anterior pero usando el GID del grupo en vez del nombre.
--home directorio → hace que directorio sea el directorio personal del usuario, en vez del directorio por defecto /home/nombre_usuario

deluser

Elimina un usuario. Existe también el comando **userdel**. Hay que tener en cuenta que no se puede borrar un usuario que esté conectado al sistema en ese momento.

Sintaxis:
deluser [opciones] usuario

Opciones:
--remove-home → borra el directorio personal del usuario (con el comando userdel esta opción sería -r).
--remove-all-files → elimina todos los ficheros que pertenezcan al usuario, además del directorio personal.

addgroup

Añade un grupo al sistema. Se obtiene el mismo resultado que si usáramos el comando siguiente: adduser --group grupo

Sintaxis:
addgroup grupo

delgroup

Elimina un grupo. Se obtiene el mismo resultado que si usáramos el comando siguiente: deluser --group grupo. Hay que tener en cuenta que no se puede borrar un grupo que esté asignado a un usuario como su grupo primario.

Sintaxis:
delgroup grupo

usermod

Modifica un usuario ya creado.

Sintaxis:
usermod [opciones] usuario

Opciones:
-d directorio [-m] → cambia el directorio personal
-m → mueve el contenido del antiguo
-g grupo → cambia el grupo primario del usuario
-u uid → cambia el uid del usuario
-l nombre → cambia el nombre del usuario
-s shell → cambia la shell

groupmod

Modifica un grupo ya creado.

Sintaxis:
groupmod [opciones] grupo

Opciones:
-n nombre → cambia el nombre del grupo
-g gid → cambia el GID del grupo

gpasswd

Añade o elimina usuario de un grupo. También se utiliza para añadir contraseñas a un grupo.

Sintaxis:
gpasswd [opciones] usuario grupo

gpasswd grupo

Opciones:
-a → añade el usuario al grupo
-d → elimina el usuario del grupo
-M → añade varios usuarios al grupo

id

Muestra la información sobre el usuario, como el UID, el nombre, los grupos a los que pertenece y los GID de los mismos.

Sintaxis:
id [opciones] [usuario]

Opciones:
-u → --user → escribe solo el UID del usuario
-n → --name → escribe el nombre del usuario, en vez del UID. Para utilizarlo debemos combinarlo con -u (id -un usuario).

groups

Muestra el nombre de todos los grupos a los que pertenece el usuario.

Sintaxis:
groups [usuario]

finger

Muestra información sobre el usuario que se le indique. Si no se especifica ninguno, muestra información sobre los usuarios conectados al sistema.

Sintaxis:
finger [usuario]

chfn

Cambia la información del usuario, que después aparecerá en el campo GECOS del fichero /etc/passwd

Sintaxis:
chfn [opciones] usuario

Opciones:
-f nombre → cambia el nombre
-r num_oficina → cambia el número de oficina
-w teléfono → cambia el número del trabajo
-h teléfono → cambia el teléfono del domicilio
-o otro → cambia otra información adicional

newgrp

Se utiliza para cambiar el grupo primario de un usuario, hasta que se vuelva a cambiar o finalicemos la sesión. Si queremos cambiar a un grupo al que no pertenezcamos, nos pide la contraseña del grupo.

Sintaxis:
newgrp grupo

chsh

Comando para cambiar la shell de un usuario. Si no se especifica ningún usuario, se entiende que es el usuario que ejecuta el comando.

Sintaxis:
chsh [opciones] [usuario]

Opciones:
-s nombre_shell → cambia a la shell indicada

Ejemplo:
Desde la terminal, crea un grupo nuevo, llamado gruponuevo. Después añade un usuario llamado usunuevo, cuyo grupo primario sea gruponuevo. Después comprueba lo realizado.

Crea otro grupo llamado grupo2. Haz que usunuevo pertenezca a grupo2, pero como grupo secundario. Comprueba lo realizado.

Al finalizar, borra los grupos y usuarios creados.

Solución:
sudo addgroup gruponuevo
sudo adduser --ingroup gruponuevo usunuevo
id usunuevo
su usunuevo (entramos como usunuevo para ver si está creado)

```
whoami
exit

sudo adduser --group grupo2
sudo adduser usunuevo grupo2
cat /etc/group (miramos el archivo de todos los grupos creados)
groups usunuevo (miramos los grupos a los que pertenece usunuevo)

su usunuevo
exit

sudo deluser --remove-home usunuevo
sudo delgroup grupo2
sudo deluser --group gruponuevo
```

1.3.3 Comandos para cambiar los ficheros y directorios de usuario y grupo

En linux, todos los archivos pertenecen a algún usuario, que es el propietario del mismo, y a un grupo al que pertenezca el usuario propietario, pero se puede cambiar el propietario y el grupo del archivo. Para ello podemos usar los siguientes comandos:

chown

Cambia el propietario del archivo. También lo podemos usar para cambiar el grupo.

Sintaxis:
chown [opciones] [propietario[:grupo]] archivo/s

Opciones:
-R → --recursive → cambia el propietario y el grupo, si se le especifica, a un árbol de directorios.

chgrp

Cambia el grupo del archivo.

Sintaxis:
chgrp [opciones] grupo archivo/s

Opciones:
-R → --recursive → cambia el grupo a un árbol de directorios.

2 PERMISOS

Al existir diferentes usuarios y grupos y como todos los archivos deben pertenecer a un usuario y a un grupo, se hace necesario protegerlos utilizando tres grupos de permisos. Los permisos del propietario del archivo, los permisos de los usuarios del grupo al que pertenece el archivo, y el resto de los usuarios.

Los permisos más frecuentes que se pueden añadir al archivo son de lectura (r), escritura (w) y ejecución (x). Si se trata de un directorio, un permiso de ejecución significa que podemos entrar en él. Si es un fichero, el permiso de ejecución tendrá significado si el fichero es ejecutable o binario, o bien contiene otros comandos, es decir, es un shell script.

2.1 GESTIÓN DE PERMISOS EN MODO TEXTO

Para comprobar los permisos de cualquier archivo, usamos el comando “ls -l”.

En modo texto tenemos los siguientes comandos para modificar los permisos de los archivos.

chmod

Cambia los permisos de un archivo.

Sintaxis:
chmod [permisos[,permisos]...] archivo/s

Permisos:
Los permisos se pueden añadir o quitar indicando lo siguiente:
chmod {a,u,g,o}{+ | - | =}{r,w,x,X,s,t} archivos

La primera letra, que será una de las que hay en el primer grupo (a,u,g,o), indica a quién se le va a cambiar el permiso: **a** → a todos, **u** → al usuario, **g** → al grupo y **o** → a los demás. Por defecto se establece la opción a.
Después podemos añadir **+** para añadir permiso o **–** para quitárselo. Alternativamente podemos usar = para establecer un permiso, pero teniendo en cuenta que quitará el resto de permisos.
Por último indicamos los permisos que queremos modificar: lectura (**r**), escritura (**w**), ejecución (**x**), ejecución en caso de que sea un directorio (**X**), y dos permisos especiales:

s (seguid bit o setgid bit) → según se aplique a un usuario o grupo, indica que el proceso pertenece al dueño del programa o a su grupo, no al que lanzó el proceso.
t (sticky bit) → aplicado a directorios hace que si un usuario tiene permiso de escritura, puede modificar los archivos, pero no los puede mover ni borrar.

Los permisos también se pueden indicar de forma numérica, como un número octal de hasta 4 dígitos (ejemplo: 432).

El primer dígito de la derecha (2) indica los permisos que se les va a dar a los demás usuarios, el segundo (3) los permisos del grupo, el tercero (4), permisos de usuario, y el cuarto, si lo hay, permisos especiales (**1** = sticky bit, **2** = setgid, **4** = setuid).

Los números indican lo siguiente:
0 = ningún permiso
1 = permiso de ejecución (x)
2 = permiso de escritura (w)
4 = permiso de lectura (r)
Para asignar combinaciones de permisos podemos sumar los números.

Ejemplo:
rwx-w-r-x → 111 010 101 (en binario) → 725 (en octal)
o bien:
rwx-w-r-x → grupo 1 → r+w+x = 4+2+1 = 7
grupo 2 → w = 2
grupo 3 → r+x = 4+1 = 5

umask

Cambia o te muestra los permisos que se asignarán por defecto.

Sintaxis:
umask [máscara]

Argumento:
La máscara de permisos tendrá distintas consecuencias dependiendo de si después de establecerla se crea un fichero o un directorio. Existen permisos base para los ficheros, que es 666, y para los directorios, que es 777.
Partiendo de esta base de permisos:
Para los directorios podemos restarle el umask a los permisos base.

Para los archivos, lo que habría que hacer sería un *and* de la negación del umask, con lo cual, en los archivos, el último bit de cada grupo (x) no se activaría nunca.

Ejemplo:
Si umask vale 033, ¿con qué permisos se crearía un archivo y un directorio?

Solución:
Para directorios:
033 → de octal a binario → 000 011 011
complementamos (cambiar 0 por 1 y 1 por 0) → 111 100 100 = rwx r-- r--
que sería 744 en octal.
O simplemente: 777 – 033 = 744

Para ficheros:
base AND (NOT umask) → 666 AND (NOT 033) → 666 AND 744
110 110 110 AND 111 100 100 = 110 100 100 = rw- r-- r--
que en octal sería 644.

Ejemplo:
Mira el valor de umask. Mira los permisos de las entradas de tu directorio personal. Cambia umask a 000. Crea un directorio llamado direcnuevo y un fichero llamado ficheronuevo. Comprueba los permisos que tienen. Cámbialos de manera que en ambos tú tengas todos los permisos activos, los de tu grupo puedan leer y ejecutar, y los otros no tengan ningún permiso activo.

Solución:
```
umask
ls -l
umask 000
mkdir direcnuevo
ls > ficheronuevo
ls -ld *nuevo
chmod 750 direcnuevo
chmod u+x,g-w,o-rw ficheronuevo
```

3 CONCEPTO DE PROCESOS

Un proceso o tarea es un programa en ejecución. El concepto de proceso y programa es diferente, ya que un mismo programa, cada vez que se ejecuta, crea un proceso diferente. Es más, puede haber distintos procesos o instancias del mismo programa ejecutándose a la vez. Como Linux es multitarea y multiusuario, puede haber muchos

procesos de distintos usuarios ejecutándose simultáneamente, y que varios procesos sean de un mismo usuario.

Un proceso además tiene un contexto, que lo diferencia de otra instancia del mismo programa en ejecución, como el usuario que lo ha lanzado (que es dueño del proceso), los permisos del proceso, si ha sido llamado por otro proceso (proceso padre)...

Para diferenciar un proceso de otro existe un número entero, PID (Process ID), que identifica a cada proceso de otro, ya que el número es diferente de un proceso a otro.

4 GESTIÓN DE PROCESOS

Para ver los procesos que se están ejecutando y para administrarlos, lo podemos hacer desde el entorno gráfico (Sistema → Administración → Monitor) o desde la terminal.

4.1 GESTIÓN DE PROCESOS EN MODO TEXTO

Para la gestión de procesos en modo texto, entramos en la terminal y podremos utilizar los comandos que veremos a continuación.

ps

Informa del estado de los procesos. Tiene varias opciones para mostrar diferentes columnas.
Entre las columnas mostradas con este comando podemos destacar las siguientes:

USER → nombre de usuario.
PID → identificador de proceso.
PPID → identificador del padre del proceso.
UID → identificador del usuario dueño del proceso.
TTY → Terminal donde se está ejecutando.
TIME → tiempo de CPU usado.
CMD → nombre del programa que inició el proceso.
RSS → tamaño de la parte residente del proceso.
SIZE → tamaño de la imagen del proceso.
NI → prioridad del proceso.
%CPU → porcentaje de CPU utilizada.
STAT → estado del proceso, que puede ser: R=ejecutándose, S=durmiendo, T=detenido,...

Sintaxis:
ps [opciones]

Opciones:
-A → para ver, de todos los procesos, las columnas PID, TTY, TIME, CMD.
-ely → estas opciones se usan unidas para ver de todos los procesos, columnas como PID, PPID, TIME, CMD, NI, RSS,…
aux → muestra todos los procesos. Estas opciones van sin guión. Muestra, entre otras, las columnas USER, PID, RSS, TTY, STAT, TIME, %CPU,…

pstree

Muestra los procesos en forma de árbol.

Sintaxis:
pstree [opciones]

Opciones:
-A → para que muestre las líneas del árbol al estilo ASCII.
-G → para que muestre las líneas al estilo VT100.
-u → muestra el propietario de los procesos.

Ejemplo:
Muestra la información sobre todos los procesos que se están ejecutando. Mira el PID de un proceso que se está ejecutando. Mira todos los procesos en forma de árbol, con las líneas estilo terminal y mostrando los propietarios. Como la lista será muy larga, pagina la salida del comando.

Solución:
ps -aux
ps -A | grep -i nombre_proceso
pstree -Gu | more

&

Se utiliza para ejecutar procesos en segundo plano, con lo cual, la terminal queda libre para seguir realizando otras tareas.

Sintaxis:
comando &

nice y **renice**

Se utilizan para ejecutar procesos con prioridad más baja o bien para bajarle la prioridad una vez que esté en ejecución, respectivamente.
La prioridad que se puede asignar a un proceso con el comando nice puede ser de -20 (máxima prioridad) a 19 (mínima). Esta prioridad se ve en la columna NI del comando ps. Después, dependiendo del valor, el sistema calcula la prioridad real del proceso, que puede tener otros valores y que se puede comprobar en la columna PRI del comando ps.

Sintaxis:
nice [opciones] comando
renice [opciones] prioridad_nueva PID_comando

Opciones:
-n número → ejecuta el comando con la prioridad que se indica en número.

jobs

Muestra los procesos asociados a una terminal. Tanto los que estén ejecutándose en segundo plano como los que estén suspendidos.
Para suspender un proceso en ejecución se utiliza la combinación de teclas CTRL+Z.

Sintaxis:
jobs

fg

De los procesos que están en segundo plano, bien ejecutándose o bien parados, se envía a ejecutarse al primer plano el proceso que se le indique.

Sintaxis:
fg %n
Donde n es el número que ocupa el proceso en la lista de procesos que nos muestra jobs. Si no se especifica ninguno, se manda al primer plano el marcado por un carácter +.

bg

De los procesos que están en segundo plano parados, se envían a ejecutarse en segundo plano el proceso que se le indique.

Sintaxis:
bg %n → donde n es el número que ocupa el proceso en la lista de procesos que nos muestra jobs.

kill

Mata el proceso que le indiquemos. También se puede utilizar para enviar otras señales a un proceso.

Sintaxis:
kill [opciones] PID

Opciones:
-9 → -SIGKILL → mata el proceso cuyo PID le enviemos como argumento.
-l → muestra todas las señales que se le pueden enviar a un proceso.
%n → donde n es el número en la lista de procesos del comando jobs.

killall

Mata todos los procesos asociados al programa que le indiquemos.

Sintaxis:
killall [opciones] nombre_proceso

Opciones:
-KILL → mata el proceso cuyo nombre le indiquemos como argumento.
-i → pide confirmación antes de eliminar cada proceso.
-l → muestra todas las señales que se le pueden enviar a un proceso.

time

Ejecuta un comando y, al terminar, muestra información sobre el tiempo que ha tardado en ejecutarse, los recursos utilizados, etc.

Sintaxis:
time comando

top

Muestra los procesos y otra información sobre el sistema, actualizada cada cierto tiempo, que por defecto es cada 3 segundos.

Sintaxis:
top

Ejemplo:
Ejecuta nano fichero4.txt Páralo y mándalo al segundo plano. Comprueba que está en segundo plano. Ejecuta nano fichero5.txt directamente en segundo plano. Vuelve a poner activo el primer proceso enviándolo al primer plano. Mata el segundo proceso. Comprueba que ya no hay procesos asociados a la terminal.

Solución:
nano fichero4.txt
CTRL+Z
jobs

nano fichero5.txt &
jobs
fg %1
CTRL+X
kill %2

EJERCICIOS

1. Si quieres que al crear un directorio tú tengas todos los permisos activados, los de tu grupo solo lo puedan leer y el resto no puedan hacer nada, ¿qué valor debería tener umask?
   Si no quieres tocar umask y ya tienes creado el directorio, cambia los permisos para que queden tal y como se ha dicho. Hazlo en modo númerico y con letras.

2. Crea dos usuarios: nuevo y nuevo2

3. Haz que te salga por pantalla solo el nombre de los usuarios que tengan una ! en la contraseña del fichero shadow.

4. Añade información personal al usuario nuevo2 y visualízala en el fichero /etc/passwd. Mírala ahora con el comando finger.

5. Crea dos ficheros a.txt y b.txt ordenados mediante sort, que contengan tres palabras cada uno. Mézclalos en otro fichero ab.txt. Muéstralo por pantalla de manera que si hay una línea repetida, no la muestre.

6. Muestra por pantalla los ficheros de tu directorio personal, de manera que muestre /, @, |, etc, dependiendo del tipo de fichero.

7. Cambia los permisos a los ficheros a.txt, b.txt y ab.txt para que solo los puedas ver tú y nadie (incluso tú) los pueda modificar o ejecutar. Hazlo de manera numérica. Añade ahora permiso de escritura para ti.

8. Ejecuta el editor nano en segundo plano, para crear f1.txt (usando &). Ejecútalo de nuevo, también en segundo plano, para crear f2.txt (usando CTRL+Z).

9. Muestra todas las tareas que tienes en segundo plano asociadas a tu terminal. Muestra ahora todos los procesos que se ejecutan en el sistema. Si quieres seguir escribiendo información en f1.txt, ¿cómo lo harías? Mata el proceso asociado a nano f2.txt

10. Copia el fichero /etc/passwd en tu subdirectorio personal y cámbiale el nombre a 'contras'. Muestra el contenido de 'contras' para que aparezcan únicamente los 5 primeros caracteres de cada línea. Úsalo de nuevo para que aparezcan únicamente los nombres de los usuarios y su uid.

11. Comprueba qué tipo de fichero es 'contras'. Comprueba si está ordenado. Si no lo está, ordena su contenido y guárdalo en 'contrasord'. Muestra la última línea de 'contrasord'.

12. Muestra todos los procesos que se ejecutan en el sistema con un comando que actualice la información cada 3 segundos. Muestra lo mismo pero con un refresco de 1 segundo.