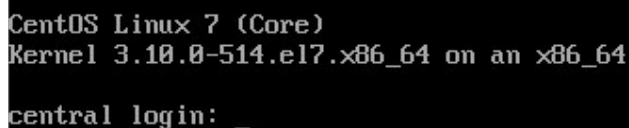


2.- USO BÁSICO DE UN SISTEMA CENTOS

2.1.- Introducción a la línea de comandos

Tras la instalación de un sistema Linux, nos encontramos la pantalla de *login*:



```
CentOS Linux 7 (Core)
Kernel 3.10.0-514.el7.x86_64 on an x86_64
central login: _
```

Pantalla de login

donde aparece en la primera línea, el nombre de la distribución (*Centos OS 7*), en la segunda línea la versión de kernel que utiliza dicha distribución (*Kernel 3.10.0, release 514, compatible el7, Enterprise Linux 7, para arquitectura x86_64*) y por último aparece el nombre del sistema o *hostname* (*central*) y la cadena de texto *login:* para indicar que introduzcamos un usuario del sistema. Una vez introducido, nos aparece la palabra *password* para indicarnos que tecleemos la contraseña del usuario.

Tras introducir la contraseña correcta, que aparece con * en lugar del texto introducido, habremos accedido al sistema con dicho usuario (también llamado hacer *login*) y aparecerá el *prompt* de la línea de comandos.

La línea de comandos o shell es la interfaz de texto que vamos a utilizar para introducir órdenes en el sistema e interactuar con él. Se utiliza por defecto la shell Bash, una versión mejorada de la shell antigua Bourne Shell.

El *prompt* de la línea de comandos, el carácter \$ para un usuario normal y # para el usuario privilegiado *root*, nos indica que está listo para recibir órdenes. Estas órdenes se dan a través de comandos.

Los comandos shell tienen la sintaxis: *<nombre_comando> <opciones> <argumentos>*, pudiendo no llevar ni opciones ni argumentos, donde:

- *<nombre_comando>*: programa a ejecutar.
- *<opciones>*: ajustan el comportamiento de un comando. Comienzan por - cuando son opciones en formato corto y -- cuando tienen formato largo.
- *<argumentos>*: sobre qué opera el comando.

Para salir de la línea de comandos, basta con teclear el comando *exit* o la tecla *Control* y la letra *D* a la vez, que se representa como *CTRL + D*.

Una cosa muy útil que tiene la shell es la posibilidad de completar comandos, subcomandos, opciones, nombres de archivos y directorios, etc., usando el tabulador. Es necesario tener instalado el paquete *bash-completion* para que funcione. Si hay varias coincidencias, al pulsar el tabulador no ocurre nada; volviéndolo a pulsar nos muestra las coincidencias.

En un sistema podemos utilizar hasta 6 consolas virtuales que soportan sesiones independientes de *login*. Si el sistema tiene entorno gráfico, éste se ejecuta en la primera consola virtual y podemos acceder a la línea de comandos ejecutando un terminal.

Las consolas virtuales están disponibles pulsando la combinación de teclas *CTRL + ALT + F<n>*, siendo *<n>* un número del 1 al 6. En el caso de estar accediendo a una máquina virtual, desde el visor de máquina virtual del Virtual Manager, podemos elegir en el menú *Send Key*, enviar estas combinaciones de teclas para acceder a las consolas virtuales del sistema.

Con el comando *tty* podremos saber en qué consola nos encontramos. El comando nos mostrará, por ejemplo, */dev/tty3*, indicando que estamos en la consola 3.

2.2.- Comandos básicos de la shell

Hay una serie de comandos muy básicos cuyo uso es muy habitual. Vamos a ver algunos de ellos con sus opciones más útiles:

- **history**: lista de los comandos que hemos ido introduciendo en la línea de comandos en el orden introducido. Al inicio de cada comando aparece un número, el número de orden en la lista del histórico. Para volver a teclear un comando de la lista, puedo teclear en la línea de comandos *!<número>* siendo *<número>* el número de orden en la lista con el que aparece el comando.

Una vez cerrado el terminal, se almacena el histórico de comandos en el *home* del usuario, dentro del archivo *.bash_history*. Hasta que se cierre el terminal, los comandos tecleados en él, estarán disponibles con el comando *history* pero no estarán escritos en el archivo.

Podemos acceder al histórico con las flechas del teclado de arriba y abajo para avanzar en el listado y volver a ejecutar un comando dado. Incluso usando la combinación de teclas *ALT + .* (la tecla *ALT* y dejándola pulsada la tecla del punto) nos aparece en la línea de comandos, la última palabra del comando anterior pudiendo seguir pulsando dicha combinación para ir accediendo a las palabras de los comandos anteriores en orden inverso.

Con la opción *-c*, borramos todas las líneas del histórico.

Por defecto se almacenan en el histórico 1000 líneas, valor configurado para todos los usuarios del sistema en la variable de entorno *HISTSIZE* que se encuentra definida en el archivo de configuración */etc/profile*.

- **date**: sin argumentos ni opciones, muestra la fecha y hora actuales. Con la opción *-d* *<tiempo>*, donde *<tiempo>* indica cuanto queremos avanzar en el tiempo, nos dice una fecha futura. Para que muestre una fecha pasada, añadiremos a lo anterior la palabra *ago*. Por ejemplo: *date -d "2 weeks"* nos dirá la fecha y hora dentro de dos semanas y *date -d "2 weeks ago"*, nos dirá la fecha de hace dos semanas. Se pueden consultar los argumentos de *<tiempo>* en el man del comando.

Una de las opciones más interesante y útiles de *date* es la opción de formato de la salida con la sintaxis: *+%<literal>* donde *<literal>* puede ser: *a* para mostrar el día de la semana abreviado, *A* el día de la semana completo, *D* para mostrar la fecha con barras, ... Consultar la página del man del comando.

- **file <archivo>**: este comando mira el contenido del archivo pasado como argumento y nos dice de qué tipo es.
- **cat <archivo>**: muestra el contenido del archivo. Si damos más de un archivo, los muestra seguidos. Si el contenido del archivo ocupa más que la pantalla, sólo veremos la parte final. Opciones interesantes:
 - *-n*: numera las líneas cuando las muestra.
 - *-b*: numera las líneas que no estén en blanco.
 - *-E*: marca el final de línea con el carácter \$.
 - *-s*: si hay varias líneas en blanco seguidas, sólo muestra una.
- **less <archivo>**: muestra el contenido del archivo dado paginando. Podemos avanzar o retroceder usando las flechas del teclado y las teclas *AvPag* y *RePag*. Incluso podemos hacer búsquedas con */<texto_buscar>*, ir al principio del archivo con *g*, ir al final con *G*.
- **head <archivo>** y **tail <archivo>**: muestran las primeras y últimas líneas respectivamente del archivo pasado como argumento. Por defecto en ambos comandos muestra 10 líneas, con la opción *-n <numero>* mostrará el número de líneas indicado.
- **wc <archivo>**: devuelve el número de líneas, palabras y caracteres del archivo, todo a la vez separado por espacios. Para que sólo devuelva una de las tres cosas, se pueden usar las opciones de *-l* para líneas, *-w* para palabras y *-c* para caracteres.

- ***which* <nombre_comando>**: muestra la ruta completa del comando que le pasamos como argumento. Cada usuario del sistema tiene definido en su entorno una variable de sistema llamada *PATH* donde están las rutas de donde se tienen que buscar los comandos. Para ejecutar un comando de sistema, no es necesario proporcionar toda la ruta a él, basta poner su nombre, la shell buscará ese comando en las rutas almacenadas en la variable *PATH*.
- ***alias* <alias>='<comando>'**: crea un alias de un comando. Útil cuando se teclea a menudo un comando con muchas opciones y argumentos. Por ejemplo, en el sistema, los usuarios tienen definido por defecto el alias *ll* que equivale a hacer *ls -l --color=auto*, un listado del directorio actual con formato largo y coloreando los archivos según su tipo para una mejor visualización.
- ***echo* <texto>**: muestra en el terminal el texto dado.

2.3.- Ayuda en la shell

Es importante saber usar con soltura la ayuda que proporciona el sistema. En una instalación de CentOS 7, los archivos de ayuda pueden llegar a suponer un 70-80% de lo que se instala en el sistema. El sistema para funcionar e interactuar con él, no necesita estos archivos, pero son imprescindibles para el administrador y los usuarios disponer de ellos. Es imposible saberse todos los comandos disponibles en un sistema y sus opciones; no se es un mal administrador del sistema por consultar mucho la ayuda.

Cada comando suele tener una opción de ayuda *-h* o *--help*, donde de forma escueta se explica su uso y las opciones principales.

En el sistema, existen tres fuentes principales para obtener ayuda: comando *man*, comando *pinfo* y la documentación extra del */usr/share/doc*.

El orden para obtener ayuda, de menos extensa a más, será: *help*, *man*, *pinfo* y */usr/share/doc*.

2.3.1.- Comando *man*

La más usada de siempre en Unix/Linux. Antiguamente eran manuales impresos y de ahí que se suele llamar a esta ayuda “las páginas del man”. Ahora se prefiere el término “temas” o “tópicos”. La ayuda del *man*, tiene 9 secciones: comandos de usuario (1), llamadas al sistema(2), funciones de librería(3), archivos especiales(4), formatos de archivos(5), juegos(6), convenciones y estándares(7), comandos privilegiados(8) y llamadas internas al kernel(9).

Puede haber el mismo tema en varias secciones, como es el caso de *passwd*, tecleando *man passwd*, nos muestra la ayuda del comando *passwd*, pero si tecleamos *man 5 passwd*, nos mostrará el formato del archivo de configuración del sistema */etc/passwd*.

El comando *man* tiene una base de datos interna que se actualiza a diario con una tarea periódica del sistema. Podemos hacer búsquedas en esa base de datos usando *man -k <texto>* o *apropos <texto>*, buscará en el nombre del tema del man y en su descripción, el texto dado. Si la búsqueda la hacemos con *-K <texto>* también buscará dentro de los temas, pero puede tardar mucho en devolvernos los resultados encontrados.

Cuando instalamos un nuevo paquete de software en el sistema, también se instalan sus temas del man pero hasta el día siguiente no se actualizará la base de datos. Funciona si ejecutamos directamente *man <nuevo_tema>*, pero al hacer una búsqueda con *man -k* o *apropos -k* no se incluirá en los resultados estos nuevos temas.

Podemos hacer que en un momento dado, p.e. tras instalar nuevos paquetes en el sistema, se actualice dicha base de datos interna: como usuario *root* ejecutaremos *mandb*.

Navegar por los temas del man es sencillo usando las teclas de *AvPag* y *RePag* del teclado, así como las flechas; además podemos hacer búsquedas dentro de un tema poniendo la barra y un texto a buscar (*/<texto>*). Con *g* y *G* vamos al principio y fin del tema y para salir del tema del *man*, usaremos la tecla *q*.

2.3.2.-Comando *pinfo*

Dentro del paquete *pinfo*, tenemos una especie de guía en línea de documentación que podemos usar ejecutando el comando *pinfo*. Está estructurado como nodos enlazados y tiene un formato más extenso y general que los temas del man. Podemos acceder a toda la documentación disponible con *pinfo* sin argumentos o pasarle como argumento el tema que queramos consultar.

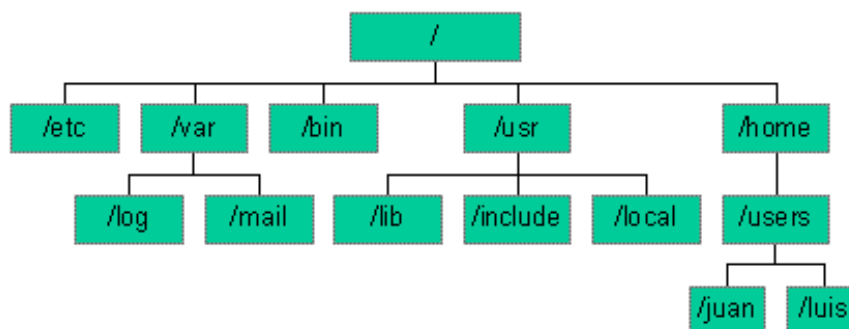
2.3.3. Documentación de */usr/share/doc*

Podremos consultar la documentación extra de un paquete de software en este directorio, si los desarrolladores la incluyeron. Es posible que con el paquete no vengan estos los archivos de documentación extra, puede haber otro paquete complementario que se llame igual que el paquete terminado en *-doc* o *-dev* que al instalarlo nos la proporcione. Por ejemplo, el paquete *kernel-doc* con documentos extras de configuración avanzada del kernel.

2.4.- Jerarquía de sistema de archivos en CentOS

Existe un estándar de jerarquía de archivos para Linux y Unix llamado FHS (**F**ilesystem **H**ierarchy **S**tandard) donde se define la estructura de archivos y directorios a utilizar por las diferentes distribuciones. En el caso de CentOS 7 y las versiones más recientes de Fedora, RHEL, OpenSuse, etc. se sigue el estándar publicado en la versión 3.0

En dicho estándar, todos los archivos y directorios aparecen bajo el directorio raíz / independientemente de si están en distintos dispositivos. La jerarquía de los directorios tiene estructura de árbol invertido siendo / el nivel más alto de la jerarquía.



Jerarquía de filesystem en Linux

2.4.1.-Directorios importantes en CentOS7

/usr: Software instalado, librerías compartidas y datos de programas de sólo lectura. Los subdirectorios más relevantes son:

/usr/bin: comandos de usuario.

/usr/sbin: comandos de administración del sistema.

/usr/local: software localmente customizado. Aquí se instala todo lo que se compile desde el código fuente.

/usr/lib y **/usr/lib64:** bibliotecas compartidas por los binarios de **/usr/bin** y **/usr/sbin**.

/usr/share/doc: documentación extra del software.

/etc: archivos de configuración específicos del sistema.

/var: datos variables específicos del sistema que persisten entre reinicios. Archivos que cambian de forma dinámica. Los subdirectorios más relevantes:

/var/tmp: archivos temporales. De forma automática los no usados (accedidos, modificados o cambiados) en 30 días son borrados.

/var/spool/mail: almacena los buzones de correo de los usuarios.

/var/log: archivos de registros de evento del sistema.

/run: datos de *runtime* de procesos como su *pid* y archivos de bloqueo. En cada reinicio se recrea el contenido de este directorio. Consolida lo que en versiones anteriores era **/var/run** y **/var/lock**.

/home: lugar donde estarán los home de los usuarios.

/root: home del superusuario del sistema, *root*.

/tmp: directorio donde se alojan los archivos temporales y donde todos los usuarios pueden escribir.

De forma automática, los archivos que no hayan sido usados (accedidos, modificados o cambiados) en 10 días son borrados.

/boot: archivos necesarios para el proceso de arranque.

/dev: archivos de dispositivo que el sistema usa para acceder al hardware.

/opt: aplicaciones de terceros.

Instalando en el sistema el paquete *man-pages*, existe un tema del man que muestra la jerarquía del sistema de archivos: *man hier*.

2.4.2.- Rutas absolutas y relativas

La ruta de un archivo indica su posición exacta en la jerarquía del sistema de archivos. Puede ser absoluta, si está indicada desde la raíz, (comienza por /) y se especifica todos los directorios y subdirectorios, separados por / hasta llegar al archivo. Una ruta relativa sin embargo, no comienza por /, sólo se da la lista de subdirectorios, separados por / hasta llegar al archivo, partiendo desde el directorio donde nos encontremos.

Por ejemplo, para el archivo *imagen1.jpg* que está en el home del usuario *alex*, en el subdirectorio de *fotos*, tendremos la ruta absoluta */home/alex/fotos/imagen1.jpg* y si estamos en el home del usuario *alex*, tendremos la ruta relativa *fotos/imagen1.jpg*.

El carácter *~* indica el home del usuario con el que hemos hecho *login*, con lo que en el ejemplo anterior, podríamos referirnos al archivo con la ruta relativa *~/fotos/imagen1.jpg*.

2.4.3.- Comandos para navegar y gestionar la jerarquía

- **cd <directorio>:** nos sitúa en el directorio pasado como argumento. Sin argumentos hace que nos situemos en el home del usuario con el que hayamos hecho *login* y sin argumentos y con:
 - **-:** nos sitúa en el directorio anterior a donde estamos ahora.
 - **..:** nos sitúa en el directorio superior al actual, el directorio padre.
 - **.<ruta>:** nos sitúa a partir del directorio actual, en la ruta que le indiquemos.
- **pwd:** nos muestra el directorio en el que estamos situados.
- **ls <directorio>:** muestra el contenido del directorio pasado o si no se proporciona, el del directorio actual. Tiene muchas opciones: **-l** para formato largo, **-a** para incluir todos los archivos incluidos los ocultos, **-R** para ver además el contenido de los subdirectorios, ...
- **stat <archivo>, stat <directorio>:** muestra la metainformación del archivo o directorio, permisos, propietario, grupo propietario, fecha del último acceso, modificación, tamaño,

- **touch <archivo>**: crea el archivo en la ruta proporcionada o si no se proporciona, en el directorio actual. Si el archivo ya existe, se actualiza su *timestamp* (su fecha de acceso, modificación y cambio).
- **mkdir <directorio>**: crea el directorio, si no se da ruta con el directorio, lo crea en el directorio actual. Con la opción *-p*, al dar una ruta, si los subdirectorios anteriores no existen, los crea a la vez que crea el directorio final de la ruta proporcionada.
- **cp <archivo> <directorio>, mv <archivo> <directorio>**: copia o mueve el archivo dado en el directorio, si con el archivo no se da una ruta, se toma del directorio actual. Se puede poner una lista de archivos pero en último lugar debe ir el directorio de destino donde se copiarán o moverán todos ellos. Usando la opción *-r* se pueden copiar o mover directorios.
- **rm <archivo>, rm -r <directorio>**: borra el archivo o directorio con su contenido.
- **rmdir <directorio>**: borra el directorio dado siempre que esté vacío.

2.5.- Redirigir la salida de los comandos

Un terminal o consola recibe los comandos por la entrada estándar (descriptor de archivo 0) o *stdin* que por defecto es el teclado y muestra la salida de estos por la salida estándar (descriptor de archivo 1) o *stdout* que es el terminal o consola, la pantalla. Además un comando puede devolver algún error, que se mostraría en el error estándar (descriptor de archivo 2) o *stderr* que, al igual que la salida estándar, por defecto es el terminal o consola. Esta entrada y salidas se pueden redirigir a archivos o pasarlos a otros comandos.

2.5.1.- Redirección a archivo

Para redireccionar la entrada estándar, utilizaremos el carácter *<*, y para redireccionar la salida estándar el carácter *>*, en ambos casos, indicando a continuación el archivo a utilizar. Para redireccionar el error estándar, utilizaremos *2>* y a continuación el archivo destino. Con la doble redirección *>>*, añadiremos al final del archivo; con la redirección simple, en el caso de que el archivo tuviese contenido, lo sobrescribimos.

Para redirigir ambas salidas, la estándar y el error, utilizaremos *> archivo 2>&1*, *>>archivo 2>&1*, *&>archivo* o *&>> archivo*.

Existe un archivo especial */dev/null* donde podemos reenviar las salidas para descartarlas.

→ Ejemplos de redirecciones usando el comando *date*:

- *date > hoy.txt*: escribe la salida del comando *date* en el archivo *hoy.txt* sobrescribiendo lo que hubiese en él. Si hubiese algún error en la ejecución del comando, lo mostrará en el terminal.
- *date >> hoy.txt*: añade al final del archivo *hoy.txt* la salida del comando *date*. Si hubiese algún error en la ejecución del comando, lo mostrará en el terminal.
- *date 2> error.txt*: escribe en el archivo *error.txt* los errores que produzca el comando *date*. La salida del comando la muestra en el terminal.
- *date 2>> error.txt*: escribe al final del archivo *error.txt* los errores que produzca el comando. La salida del comando la muestra en el terminal.
- *date >> hoy_con_error.txt 2>&1*: escribe al final del archivo *hoy_con_error.txt* la salida estándar del comando y los errores que se produzcan, no mostrando nada en el terminal.
- *date 2> /dev/null*: muestra la salida del comando en el terminal y descarta los errores.

Usando redirecciones, tenemos una forma rápida de crear un archivo desde el terminal:

```
cat > hola.txt << EOF
hola
que
tal
EOF
```

2.5.2.- Envío a otro comando

La salida estándar de un comando se puede enviar a la entrada estándar de otro usando tuberías (en inglés, pipes) utilizando el carácter | separando los comandos. Se pueden concatenar varias tuberías en la misma línea. El ejemplo más típico es cuando tenemos una salida muy larga de un comando que no cabe en la pantalla del terminal y se usa una tubería con el comando *less* para que pague la salida, p.e., *ls -l /usr/bin | less*.

Un comando que nos permite combinar redirecciones con tuberías es el comando *tee*, que permite enviar la salida de un comando a la entrada de otro comando y a la vez hacer una redirección, bien a la pantalla o a un archivo, p.e., *ls -l /usr/bin | tee listado_comandos.txt | less* que mostrará el contenido del directorio */usr/bin* por el terminal y además lo escribe en el archivo *listado_comandos.txt*.

2.6.- Ajuste de la fecha, hora y zona horaria del sistema

Es importante en un sistema que esté bien configurado la zona horaria, fecha y hora. Todos los sistemas de una red deben tener los mismos valores para que funcionen de forma sincronizada, bien para los accesos remotos, a la hora de analizar los archivos de *logs*, ...

En la instalación, se establecen estos parámetros pero puede ser necesario cambiarlos.

Un sistema Linux tiene dos relojes:

- el reloj hardware llamado RTC (**R**ea**T** **T**ime **C**lock) y que es independiente del reloj que lleva el sistema operativo ya que sigue en funcionamiento incluso cuando el sistema operativo no lo está. El sistema lleva una pequeña batería que lo mantiene en funcionamiento.
- El reloj de sistema o software, mantenido por el kernel de Linux que toma su valor inicial del RTC pero que luego es independiente de este.

Además se puede utilizar el protocolo NTP para sincronizar el reloj del sistema con otro sistema.

NTP (**N**etwork **T**ime **P**rotocol) es un estándar para sincronizar los relojes de los sistemas informáticos a través de enrutamiento de paquetes usando el protocolo UDP por el puerto 123. Se obtiene la información de los servidores públicos de Internet, como por ejemplo, los del proyecto NTP Pool. Los servidores NTP tienen una estructura jerárquica, llamada *stratio*, donde, en la posición más alta están los servidores sincronizados con un reloj atómico o GPS. Estos servidores tienen *stratum* 0, los que se conectan a estos tienen *stratum* 1 y así sucesivamente.

Con el comando *timedatectl* se ven las configuraciones actuales del sistema: fecha y hora locales, zona horaria, ... y si está funcionando con el protocolo NTP y si está el sistema sincronizado.

2.6.1.- Ajuste de la zona horaria

Para ver las zonas horarias disponibles, ejecutaremos el comando *timedatectl* con el subcomando *list-timezones*. Este comando, consulta los directorios y archivos de */usr/share/zoneinfo*. La lista es muy extensa y en ocasiones puede ser difícil saber qué zona horaria es la que se debe seleccionar. Para ayudarnos, existe el comando *tzselect* que hace una serie de preguntas: el continente donde nos encontramos, el país, la región dentro del país,.. y según las respuestas dadas, proporciona la zona horaria que se debe usar.

Para configurar la zona horaria del sistema, ejecutamos el comando:

```
timedatectl set-timezone <zona_horaria>
```

siendo *<zona_horaria>* la proporcionada por el comando *tzselect* mencionado antes. Esto hace un enlace simbólico del archivo de zona de */usr/share/zoneinfo* en */etc/localtime*.

2.6.2.- Ajuste de la fecha y la hora

La fecha y hora actuales del sistema se pueden consultar con:

```
timedatectl    ó    timedatectl status
```

Para hacer cambios, usando el comando:

```
timedatectl set-time <fecha> <hora>
```

cambiamos la fecha y la hora del sistema operativo; se puede omitir uno de los dos para cambiar sólo el otro; *<fecha>* es la fecha en formato *año-mes-día* y *<hora>* es la hora con formato *hora:minutos:segundos*.

También se puede hacer que se sincronice o no por NTP usando:

```
timedatectl set-ntp <valor>
```

con *<valor>* a *true* o *false* respectivamente.

Para cambiar la fecha y hora del reloj hardware, tenemos el comando *hwclock*. Sin parámetros muestra la configurada actualmente y con:

```
hwclock -set --date "<fecha> <hora>"
```

establece la fecha hora del sistema con *<fecha>* usando el formato *año-mes-día* y *<hora>* usando el formato *<hora>:<minutos>:<segundos>*.

Además:

hwclock -systohc: establece la hora hardware tomándola de la del sistema operativo.

hwclock -hctosys: establece la fecha y hora del sistema operativo tomandola del hardware.

2.6.3.- Configurar Chronyd

El NTP en CentOS 7 ha dejado de estar controlado por el demonio *ntpd* y pasa a estarlo por el servicio de *chronyd*, que mantiene el reloj del sistema sincronizándolo con los servidores NTP.

Por defecto se usan los servidores del proyecto NTP Pool pero existen otros servidores públicos que se pueden usar o uno propio interno de la organización.

Para ver los datos de sincronización del sistema con los servidores configurados en Chrony ejecutaremos:

```
chronyc sources -v
```

la salida del comando es bastante explicativa.

En el archivo de configuración de Chrony */etc/chrony.conf* se puede configurar cuales son los servidores NTP que se utilizarán para la sincronización. Se configuran en las líneas con formato:

```
server <servidor_ntp> iburst
```

donde por defecto en *<servidor_ntp>* aparecen los servidores del proyecto *NTP Pool*.

Tras hacer cambios en el archivo, hay que reiniciar el servicio con `systemctl restart chronyd`.

Existe una guía bastante completa de Chrony en el sistema en `/usr/share/doc/chrony-*/chrony.txt.gz`.

2.7.- Caso práctico

Con el usuario *admin* existente en **server1**, vamos a crear en su home el directorio *musica* donde se crearán los archivos *tema1.mp3*, *tema2.mp3*, *video1.avi*, *video2.avi*. A continuación, se creará el directorio *videos* en el home de *admin* a donde se moverán los archivos *video1.avi* y *video2.avi*.

Se creará un archivo *indice.txt* en el home de *admin* donde en la primera línea aparezca el texto “*INDICE*”, en la segunda la fecha de hoy con el formato día-mes-año y a continuación el contenido de ambos directorios *musica* y *videos* usando la salida del comando *ls*.

Cambiar el servidor NTP de **server1** al servidor NTP de stratum 1 *hora.roa.es*.

RESOLUCIÓN

- Entramos al sistema **server1** con el usuario *admin* y creamos los archivos pedidos:

```
[admin@server1 ~]$ mkdir musica
[admin@server1 ~]$ cd musica
[admin@server1 musica]$ touch tema1.mp3 tema2.mp3
[admin@server1 musica]$ touch video1.avi video2.avi
[admin@server1 musica]$ mkdir ~/videos
[admin@server1 musica]$ mv video1.avi video2.avi ../videos
[admin@server1 musica]$ echo 'INDICE' > ~/indice.txt
[admin@server1 musica]$ date +%d-%m-%Y >> ~/indice.txt
[admin@server1 musica]$ ls ~/musica >> ~/indice.txt
[admin@server1 musica]$ ls ~/videos >> ~/indice.txt
[admin@server1 musica]$ cat ~/indice.txt

INDICE
03-09-2017
tema1.mp3
tema2.mp3
video1.avi
video2.avi
```

- Cambiamos el servidor NTP en **server1**, como usuario *root*:

```
[root@server1 ~]# chronyc sources -v
210 Number of sources = 4
.-- Source mode '^' = server, '=' = peer, '#' = local clock.
/ .- Source state '*' = current synced, '+' = combined , '-' = not combined,
```

Caso práctico de configuración de sistemas CentOS7

/	'?' = unreachable, 'x' = time may be in error, '-' = time too variable.									
	. - xxxx [yyyy] +/- zzzz									
	Reachability register (octal) - . xxxx = adjusted offset,									
	Log2(Polling interval) --. yyyy = measured offset,									
	\ zzzz = estimated error.									
	\									
MS Name/IP address	Stratum	Poll	Reach	LastRx	Last sample					
=====										
^ ntp.redimadrid.es	2	10	377	97	+69us[+69us]	+/-	49ms			
^ 213.251.52.234	2	10	377	123	-967us[-967us]	+/-	57ms			
^ dnscache-madrid.ntt.eu	2	9	377	196	+6531us[+6531us]	+/-	61ms			
^* i2t15.i2t.ehu.eus	1	10	377	791	+138us[-503us]	+/-	8742us			

```
[root@server1 ~]# less /etc/chrony.conf
```

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
server 3.centos.pool.ntp.org iburst
```

Vemos que aparecen los servidores públicos del proyecto CentOS. Dejamos sólo una línea server con *hora.roa.es* y reiniciamos el servicio:

```
[root@server1 ~]# systemctl restart chronyd
```

```
[root@server1 ~]# chronyc sources -v
```

```

210 Number of sources = 1
.-- Source mode '^' = server, '=' = peer, '#' = local clock.
/ .- Source state '*' = current synced, '+' = combined , '-' = not combined,
| / '?' = unreachable, 'x' = time may be in error, '~' = time too variable.
| |                                     .- xxxx [ yyyy ] +/- zzzz
| |                                     | xxxx = adjusted offset,
| |         Reachability register (octal) -.      | yyyy = measured offset,
| |         Log2(Polling interval) --.    |      | zzzz = estimated error.
| |                                     \   | |
| |                                     |   |
| |                                     \   |
MS Name/IP address          Stratum Poll Reach LastRx Last sample
=====
^* hora.roa.es              1 6 27 2 -27us[-107us] +/- 12ms

```

3. GESTIÓN DE USUARIOS Y GRUPOS LOCALES



3.1.- Introducción

Los sistemas Unix/Linux son sistemas multiusuario que aseguran el uso del sistema de forma segura y ordenada. Para poder utilizar un sistema Unix/Linux es necesario disponer de una cuenta de usuario que se compone principalmente de un nombre de usuario o *login* y una contraseña o *password*. Las cuentas de usuario son creadas por el usuario administrador *root*. Otros componentes de una cuenta de usuario son su grupo primario (grupo obligatorio al que un usuario debe pertenecer), grupos secundarios (grupos extra al que el usuario pertenece), una ruta donde el usuario almacena sus documentos, normalmente un subdirectorio dentro de */home*, un intérprete de comandos o *shell* que le permitirá ejecutar aplicaciones y un *uid* (identificación de usuario con el que el sistema identifica al usuario de forma única).

Cuando un usuario ejecuta una aplicación, el sistema la carga en memoria y la ejecuta, llamándose proceso. Cada proceso en el sistema se ejecuta como un usuario en concreto y cada archivo o directorio pertenece a un usuario. El acceso a archivos y directorios está restringido por usuario y el usuario asociado a un proceso determina qué archivos y directorios son accesibles para el proceso.

Con el comando *ps aux* veremos los procesos en ejecución del sistema con el usuario que lo ejecutó. Con el comando *id* sin parámetros vemos la información del usuario con el que estamos conectados (*uid*, *gid* y grupos a los que pertenece) y con *id <usuario>* la información de ese usuario en concreto.

3.2.- Usuarios locales

En el proceso de instalación estándar de las distribuciones Linux como CentOS 7, se crea el usuario *root* y se le da contraseña. En este proceso existe la posibilidad de crear más usuarios pero se suele hacer ya con el sistema en funcionamiento, desde el usuario *root* usando el comando *useradd*.

3.2.1.- Tipos de usuarios locales:

- **Usuario root:** también llamado superusuario o usuario administrador. Su UID es 0 y es la única cuenta con privilegios totales en el sistema. Tiene acceso total a archivos, directorios y procesos independientemente de los permisos que estos tengan. Puede instalar software, ejecutar tareas de mantenimiento del sistema, ...

- **Usuarios de sistema:** son usuarios utilizados para ejecutar ciertos procesos de sistema, no tienen todos los privilegios de root pero pueden asumir algunos. No suelen tener home ni contraseña de acceso ya que no inician sesión en el sistema. Se suelen crear en la instalación del sistema o de los paquetes. En CentOS 7 tienen UID de 1-200 para los usuarios que se asignan de forma estática o de 201-999 para procesos de sistema que no tienen archivos en los sistemas de archivos. Usuarios de sistema: *bin, apache, daemon,...*
- **Usuarios regulares:** usuarios individuales. Sólo tienen privilegios totales en su home que está situado bajo /home. En CentOS 7 sus UIDs tienen valores a partir de 1000.

En el archivo */etc/login.defs* están definidos estos rangos de usuarios que se pueden modificar.

3.2.2.- Archivo */etc/passwd*

Por defecto, el sistema usa el archivo plano */etc/passwd* para almacenar la información de los usuarios locales. Este archivo almacena en una línea la información de un usuario usando 7 campos separados por el carácter dos puntos (:) con el formato:

`<username>:<password>:<uid>:<gid>:<gecos>:<home_dir>:<shell>`

donde:

`<username>`: es el nombre del usuario en el sistema. No debe contener mayúsculas.

`<password>`: password del usuario encriptada. Actualmente en este campo aparece una x ya que las password encriptadas están en el archivo */etc/shadow* junto con su caducidad. Si aparece un asterisco, el usuario no se podrá logar en el sistema y si aparece vacío, el usuario podrá entrar al sistema sin necesidad de introducir password.

`<uid>`: identificador único del usuario en el sistema.

`<gid>`: identificador único del grupo primario al que pertenece el usuario.

`<gecos>`: campo optativo usado para almacenar comentarios que suele conteter el nombre completo del usuario.

`<home_dir>`: directorio home del usuario. Si es un usuario de sistema, puede estar vacío.

`<shell>`: intérprete de comandos a utilizar por el usuario, p.e. */bin/bash*. Si aparece */bin/nologin* el usuario no podrá hacer *login* en el sistema.

Este archivo tiene permiso de lectura para todos los usuarios, ya que hay muchos comandos que lo consultan pero sólo tiene permiso de escritura para el usuario root.

No sería recomendable modificar este archivo con un editor de textos, salvo utilizando el comando *vipw* que utiliza bloqueos para que no se puedan hacer modificaciones en el archivo si lo tenemos abierto.

También los comandos *useradd*, *userdel* y *usermod* permiten hacer las modificaciones necesarias. Existe el comando *pwck* que verifica la integridad del archivo */etc/passwd* y */etc/shadow*.

3.2.3.- Crear usuarios

Como usuario root utilizaremos el comando ***useradd* <opciones> <username>** que añade una línea en el archivo */etc/passwd* para el usuario con *login <username>*, añade una línea en el archivo */etc/shadow* y se modifica el archivo */etc/group*. Se pueden añadir las opciones de:

- c <comentario>, --comment <comentario>: para poner valor al campo *gecos*.
- g <grupo>, --gid <grupo>: para especificar un grupo primario en concreto con *gid* o nombre de grupo. El grupo debe existir previamente. Si no se especifica esta opción, por defecto se creará un grupo con el mismo nombre que el *username* y será el grupo primario del usuario.
- b <base_dir>, --base-dir <base_dir>: se especifica el directorio base del directorio home del usuario, si no se añade la opción -m, el directorio debe existir. Si no se usa esta opción, por defecto es */home*.
- d <home_dir>, --home-dir <home_dir>: directorio home del usuario bajo el <base_dir>. No debe existir previamente ya que el comando lo crea.
- e <yyyy-mm-dd>, --expiredate <yyyy-mm-dd>: fecha de expiración de la cuenta.
- G <nombre_grupo1>,<nombre_grupo2>,..., --groups <nombre_grupo1>,<nombre_grupo2>,...: lista de grupos suplementarios a los que pertenece el usuario que deben existir previamente.
- r, --system: crea un usuario de sistema, dándole un *uid* por debajo de 1000. No crea el home del usuario salvo que se utilice la opción -m.
- s <shell>, --shell <shell>: shell que utilizará el usuario, p.e. */bin/bash*, */bin/sh*, */bin/nologin*.
- u <num>, --uid <num>: si se quiere asignar un *uid* en concreto al usuario.

NOTA: En el archivo */etc/default/useradd* están especificadas los valores por defecto de los diferentes campos si no se especifican con la opción correspondiente.

3.2.4.- Modificar usuarios

Con el usuario *root* y el comando ***usermod* <opciones> <username>** podemos modificar las propiedades del usuario <username> almacenadas en */etc/passwd* y */etc/group* y bloquear/desbloquear usuarios. Opciones más comunes:

- c <comentario>, --comment <comentario>: para modificar el valor al campo *gecos*.

- g <grupo>, --gid <grupo>: para modificar el grupo primario en concreto. El grupo debe existir previamente.
- G <grupo1>,<grupo2>,..., --groups <grupo1>,<grupo2>,...: Para sustituir los grupos suplementarios del usuario. Si lo que queremos es añadir grupos secundarios al usuario a los que ya tiene, hay que añadir la opción -a. Esto hace modificaciones en */etc/group*.
- d <home_dir>, --home-dir <home_dir>: modifica el directorio home del usuario. De existir previamente salvo que utilicemos además la opción -m que mueve el contenido del home actual al nuevo home y si no existe lo crea.
- l <nuevo_username>, --login <nuevo_username>: modifica el *username* del usuario. Hay que tener precaución con esta opción ya que no modifica el nombre del directorio home del usuario ni el de correo.
- s <shell>, --shell <shell>: modifica la shell que utilizará el usuario.
- L, --lock: bloquea la cuenta del usuario. Esto modifica el archivo */etc/shadow* añadiendo el carácter ! al principio del segundo campo.
- U, --unlock: desbloquea la cuenta del usuario. Esto modifica el archivo */etc/shadow* quitando el carácter ! al principio del segundo campo.

3.2.5.- Borrar usuarios

Con el usuario *root* y el comando ***userdel*** <username> se elimina el usuario del sistema. Esto implica que se borra la línea correspondiente del archivo */etc/passwd* y del archivo */etc/shadow* y se elimina dicho usuario de los grupos en */etc/group*. No se elimina el home del usuario salvo que se añada la opción -r o --remove.

Los archivos y directorios pertenecientes al usuario que estén en otros sistemas de archivos, permanecen en el sistema pero al haber desaparecido el usuario, en lugar de aparecer en su owner el nombre del usuario y en el grupo el nombre del grupo, aparece el *uid* y *gid* respectivamente. Dejar estos archivos sin eliminar del sistema, es peligroso ya que si posteriormente creamos un usuario con ese mismo *uid* y un grupo con ese *gid*, los archivos de forma automática se les asignarán.

Tal y como se indica en el man del comando *userdel*, es interesante modificar en */etc/login.defs* la variable *USERDEL_CMD* para que apunte a un script que se ejecute al borrar un usuario del sistema y que dicho script haga una búsqueda por todos los sistemas de archivos del sistema de los archivos y directorios del usuario y los elimine además de eliminar también cualquier tarea *at* (*find /var/spool/cron/atjobs -name "[^.]*" -type f -user <usuario> -delete \;*), tarea *cron* (*crontab -r -u <usuario>*) y de impresión (*lprm <usuario>*) que estén pendiente.

3.2.6.- Caso práctico

En **server1** vamos a modificar que los usuarios regulares comiencen en 1100 en lugar de 1000 al igual que en los grupos regulares; crear los usuarios regulares con *username* *visitor1* y *visitor2* y crear un usuario de sistema con *username* *jboss* que no pueda hacer *login* en el sistema y con *home* en */opt/jboss*.

Luego se borrará el usuario *visitor2* eliminando su *home*.

RESOLUCIÓN:

- Comenzamos modificando el valor en el que empezarán a asignarse usuarios y grupos regulares en */etc/login.defs*:

```
[root@server1 ~]# sed -i 's/UID_MIN          1000/UID_MIN          1100/g' /etc/login.defs
[root@server1 ~]# sed -i 's/GID_MIN          1000/GID_MIN          1100/g' /etc/login.defs
[root@server1 ~]# useradd -c 'Usuario visitante 1' visitor1
[root@server1 ~]# useradd -c 'Usuario visitante 2' visitor2
[root@server1 ~]# tail -2 /etc/passwd
visitor1:x:1100:1100:Usuario visitante 1:/home/visitor1:/bin/bash
visitor2:x:1101:1101:Usuario visitante 2:/home/visitor2:/bin/bash
```

- Creamos el usuario de sistema *jboss*:

```
[root@server1 ~]# useradd -r -c 'Usuario de sistema para JBoss' -s /bin/nologin -b /opt jboss
[root@server1 ~]# tail -2 /etc/passwd
visitor2:x:1101:1101:Usuario visitante 2:/home/visitor2:/bin/bash
jboss:x:995:993:Usuario de sistema para JBoss:/opt/jboss:/bin/nologin
```

- Elimino el usuario *visitor2*:

```
[root@server1 ~]# userdel -r visitor2
[root@server1 ~]# tail -2 /etc/passwd
visitor1:x:1100:1100:Usuario visitante 1:/home/visitor1:/bin/bash
jboss:x:995:993:Usuario de sistema para JBoss:/opt/jboss:/bin/nologin
```

3.3.- Gestionar contraseñas

Con el comando **passwd** cualquier usuario puede modificar su propia contraseña de acceso al sistema. Sólo el usuario *root* podrá modificar las contraseñas de otros usuarios usando **passwd** *<usuario>* siendo *<usuario>* el usuario al que *root* quiere cambiarle la contraseña.

En los usuarios regulares, existen unas restricciones para elegir contraseña: al menos 8 caracteres, no puede estar basada en una palabra existente en el diccionario, no utilizar el *username*, no usar

una contraseña anterior, ...

Estas restricciones están definidas en el archivo `/etc/login/defs`. El usuario `root` se salta estas restricciones pudiendo asignar cualquier contraseña a los usuarios.

Este comando pide dos veces por la entrada estándar la introducción de la nueva contraseña. Existe una opción del comando, utilizada para scripts, que permite pasarle la contraseña por línea de comandos `--stdin` pero su uso está totalmente desaconsejado ya que la contraseña se ve en texto claro en el terminal y se queda almacenada en el histórico del usuario. Su uso sería: `echo <nueva_pass> | passwd --stdin <usuario>`.

3.3.1.- Archivo `/etc/shadow`

Las contraseñas se almacenan encriptadas en el archivo `/etc/shadow`. A este archivo sólo el usuario `root` tiene acceso ya que no tiene permisos para hacer absolutamente nada en él.

Por cada usuario existe una línea en este archivo, 9 campos separados por el carácter `:`, de la forma: `<username>:<password>:<ultimo_cambio>:<min>:<max>:<aviso>:<inactiva>:<expirada>:<blanco>` donde:

`<username>`: nombre de usuario en el sistema

`<password>`: contraseña encriptada. Si el primer carácter es `!` significa que el usuario está bloqueado.

Si aparece `!!` significa que el usuario no tiene contraseña asignada y si aparece `*` significa que el usuario es un usuario *no-login* (usuarios de sistema utilizados para procesos que no se loguearán nunca), en ambos casos, ese usuario no podrá hacer *login* en el sistema.

`<ultimo_cambio>`: días transcurridos desde el 1 de enero de 1970 hasta que se cambió la contraseña por última vez. Si aparece un `0`, significa que el usuario debe cambiar la contraseña en el siguiente *login*. Si está vacío, significa que la funcionalidad de caducidad de la contraseña está desactivada.

`<min>`: número mínimo de días transcurridos desde que se puso la última contraseña para que pueda ser cambiada. Si aparece un `0`, no hay número mínimo de días.

`<max>`: número máximo de días antes de que una contraseña tenga que cambiarse obligatoriamente. Trascurridos estos días, obligará a un cambio de contraseña en el siguiente *login*. Si aparece un `-1`, no hay número máximo de días. Si *max* es menor que *min*, el usuario no podrá cambiar la contraseña.

`<aviso>`: días antes de que una contraseña expire en los cuales se avisa al usuario cuando hace *login*. Si aparece un `0`, no hay aviso.

<inactiva>: número de días tras la expiración de una contraseña antes de que se bloquee. En este periodo se pueden entrar con el usuario y la contraseña pero obliga a cambiarla. Puede estar vacío.

<expirada>: días pasados desde el 1 de enero de 1970 hasta la fecha de expiración de la cuenta. Si está vacío significa que la cuenta nunca expira.

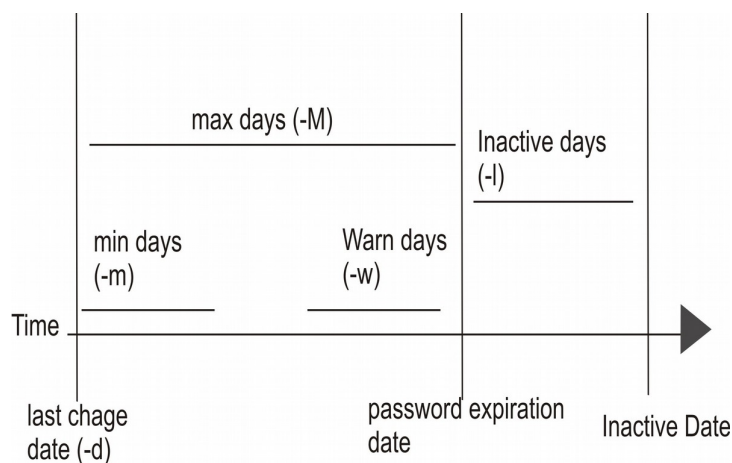
<blanco>: campo en blanco reservado para usos futuros.

Los usuarios a los que no se les ha asignado contraseña nunca, el campo *password* aparece con !!.

No sería recomendable modificar este archivo con un editor de textos, salvo utilizando el comando *vipw -s* que utiliza bloqueos para que no se puedan hacer modificaciones en el archivo si lo tenemos abierto. También los comandos *passwd* y *chage* permiten hacer las modificaciones necesarias. Existe el comando *pwck* que verifica la integridad del archivo */etc/passwd* y */etc/shadow*.

3.3.2.- Gestionar caducidad de la contraseña de un usuario

En el siguiente diagrama se ven los parámetros de caducidad que se ajustan con el comando *chage*.



Parámetros del comando *chage*

Con el comando ***chage* <opciones> <username>** podemos ajustar la caducidad de la contraseña de un usuario. Sólo puede ser ejecutado como *root*. Las opciones a usar:

-d <dia>, --last-day <dia>: días desde 1 de enero de 1970 cuando la contraseña fue cambiada. Se puede poner una fecha con el formato *yyyy-mm-dd* o poner 0 para obligar a cambiar la contraseña en el siguiente *login*.

-l: lista la configuración de caducidad de contraseña actual del usuario.

-E <dia>, - -expiredate <dia>: días desde 1 de enero de 1970 cuando expirará la contraseña. Se puede poner una fecha con el formato *yyyy-mm-dd*. Si se pone -1, borrará la fecha de

expiración de la cuenta.

-I <días>, **--inactive** <días>: días de inactividad tras caducar la contraseña (pasados los max-days).

-m <días>, **--mindays** <días>: días a transcurrir desde el último cambio de contraseña para poder volver a cambiarla.

-M <días>, **--maxdays** <días>: días que puede durar una contraseña.

-W <días>, **--warndays** <días>: días antes de que caduque la contraseña que avisa al usuario al hacer *login*.

3.3.3.- Caso práctico

En **server1**, al usuario *visitor1* creado anteriormente, se le asigna la contraseña de acceso al sistema *passvisitor*. Obligamos a que la primera vez que el usuario haga *login*, cambie la contraseña, que debe cambiar cada 60 días, dando un aviso tres días antes y permitiendo un periodo de gracia de cinco días. El usuario podrá cambiar la contraseña siempre que quiera y la cuenta del usuario expirará el 31 de diciembre de 2017.

RESOLUCIÓN:

- Asigno contraseña por línea de comandos al usuario *visitor1*:

```
[root@server1 ~]# echo 'passvisitor' | passwd visitor1 --stdin
```

- Veo la caducidad de la contraseña que tiene por defecto el usuario antes de hacer cambios:

```
[root@server1 ~]# chage -l visitor1
```

```
Last password change           : Aug 21, 2017
Password expires                : never
Password inactive               : never
Account expires                 : never
Minimum number of days between password change : 0
Maximum number of days between password change : 99999
Number of days of warning before password expires : 7
```

- El usuario debe cambiar la contraseña en el primer *login*, cambiarla cada 60 días, con aviso tres días antes de caducar, con 5 días de gracia y que la fecha de expiración de la cuenta sea 31 de diciembre de 2017:

```
[root@server1 ~]# chage -d 0 -M 60 -W 3 -I 5 -E 2017-12-31 visitor1
```

- Veo la caducidad de la contraseña tras los cambios:

```
[root@server1 ~]# chage -l visitor1
```

```
Last password change           : password must be changed
```

Password expires	: password must be changed
Password inactive	: password must be changed
Account expires	: Dec 31, 2017
Minimum number of days between password change	: 0
Maximum number of days between password change	: 60
Number of days of warning before password expires	: 3

Una vez que el usuario ha entrado al sistema y ha establecido una nueva contraseña:

```
[admin@server1 ~]$ su - visitor1
```

```
Password: ← passvisitor
```

```
You are required to change your password immediately (root enforced)
```

```
Changing password for visitor1.
```

```
(current) UNIX password: ← passvisitor
```

```
New password: ← Vi23Valencia
```

```
Retype new password: ← Vi23Valencia
```

la caducidad de la contraseña queda así:

```
[root@server1 ~]# chage -l visitor1
```

```
Last password change : Aug 21, 2017
```

```
Password expires : Oct 20, 2017
```

```
Password inactive : Oct 25, 2017
```

```
Account expires : Dec 31, 2017
```

```
Minimum number of days between password change : 0
```

```
Maximum number of days between password change : 60
```

```
Number of days of warning before password expires : 3
```

Y la línea correspondiente en el archivo `/etc/shadow`:

```
[root@server1 ~]# cat /etc/shadow | grep visitor1
```

```
visitor1:$6$OVk.0nh9$OJYYSs.75WQQzoCcUYTW9QmS0Z9sgpCkl5eAaOfiT8VqHuXx/iOS5hz7qXairo4qAO3O73QiGLtkl.4.XDekm1:17399:0:60:3:5:17531:
```

3.4.- Grupos locales

Un grupo debe existir antes de que un usuario pueda ser añadido a él, salvo el grupo primario del usuario que se crea por defecto con el mismo nombre que el del usuario, al usar `useradd` sin usar la opción `-g`.

Con el comando `groupmems -g <nombre_grupo> -l` obtengo una lista con los usuarios que pertenecen a ese grupo.

3.4.1.- Archivo `/etc/group`

Al igual que el archivo `/etc/passwd` almacena la información de los usuarios locales, el archivo `/etc/group` almacena la información de los grupos locales del sistema.

La información de un grupo es una línea de 4 campos separados por el carácter dos puntos (:) con el formato:

```
<groupname>:<password>:<gid>:<username1>,<username2>,...., <usernameN>
```

donde:

`<groupname>`: es el nombre del grupo en el sistema. No debe contener mayúsculas.

`<password>`: contraseña del grupo encriptada. No se suele utilizar, aparece vacío.

`<gid>`: identificador único del grupo en el sistema.

`<username1>,....`: lista de usuarios que pertenecen al grupo separados por comas. Puede estar vacía bien porque es un grupo primario o porque aún no se le han asignado usuarios.

Este archivo tiene permiso de lectura para todos los usuarios, ya que hay muchos comandos que lo consultan pero sólo tiene permiso de escritura para el usuario *root*.

Al igual que ocurre con el archivo `/etc/passwd`, no es recomendable modificar este archivo con un editor de textos, salvo utilizando el comando *vigr* que utiliza bloqueos para que no se puedan hacer modificaciones en el archivo si lo tenemos abierto. También los comandos *groupadd*, *groupdel* y *groupmod* permiten hacer las modificaciones necesarias. Con el comando *grpck* chequeamos la integridad del archivo `/etc/group`.

3.4.2.- Crear grupos

Con el comando ***groupadd*** `<opciones>` `<nombre_grupo>` creamos un nuevo grupo secundario en el sistema. Las opciones usadas:

`-g <gid>`: dar un gid en concreto, opción nada recomendada, es mejor dejar que el sistema los asigne de forma automática para que no haya colisiones en los *gid*.

`-r`: para crear un grupo de sistema que tendrá *gid* por debajo de 1000.

Una vez creado el grupo, añadir un usuario al grupo con el comando *useradd -aG <grupo>* `<usuario>`. Importante el uso de la opción `-a` junto con `-G` para que añada a los grupos secundarios a los que ya pertenece el usuario, este nuevo grupo. Si sólo usamos `-G`, eliminamos el usuario de TODOS los grupos secundarios a los que pertenece y lo añadimos al nuevo grupo. Esto no afecta a la pertenencia al grupo primario.

3.4.3.- Modificar grupos

Con el comando *groupmod* <opciones> <nombre_grupo> modifico las propiedades del grupo como pueden ser nombre del grupo o *gid*. Opciones usadas:

-g <gid>: cambio el *gid* del grupo. Importante saber que no se hacen comprobaciones en el archivo */etc/login.defs* para los parámetros *GID_MIN*, *GID_MAX*, *SYS_GID_MIN*, *SYS_GID_MAX* con lo que es posible que estemos infringiendo esos rangos establecidos. Los archivos/directorios que tuviesen el antiguo *gid*, deben ser cambiados de forma manual al nuevo *gid*.

-n <nuevo_nombre>: para cambiar el nombre de grupo.

3.4.4.- Borrar grupos

Se utiliza el comando *groupdel* <nombre_grupo>. El grupo no será borrado si es un grupo primario de algún usuario. Al igual que con el comando *userdel*, hay que asegurarse de que en los sistemas de archivos del sistema no existen archivos que pertenezcan al grupo.

3.4.5.- Caso práctico

En **server1** se van a crear una serie de grupos para los distintos departamentos de la empresa: *desarrollo*, *sistemas*, *redes* y *seguridad*. Existe un director por departamento que además de pertenecer al grupo de su departamento, pertenece al grupo *directiva*.

También se van a crear los usuarios pertenecientes a los departamentos, usando como *username* la inicial del nombre seguida del apellido. La distribución de los usuarios por departamentos, siendo el primer usuario que aparece el director, es:

- *desarrollo*: Ada Lovelace, Grace Hopper.
- *redes*: Hedy Lamarr, Tim Berners, Robert Kahn.
- *seguridad*: Alan Turing y Tatu Ylonen.
- *sistemas*: Linus Torvalds, Dennis Ritchie.

A todos los usuarios se les dará la contraseña de *changeme* que deberán cambiar en el primer *login*.

RESOLUCIÓN:

Creamos los grupos de los departamentos utilizando un bucle *for* de Bash:

```
[root@server1 ~]# for dpto in {desarrollo,redes,seguridad,sistemas,directiva};do
                                groupadd $dpto;done

[root@server1 ~]# tail -5 /etc/group
desarrollo:x:1101:
redes:x:1102:
```


seguridad:x:1103:

sistemas:x:1104:

directiva:x:1105:

Creamos los usuarios de los departamentos:

```
[root@server1 ~]# useradd -G desarrollo -c 'Ada Lovelace' alovelace
```

```
[root@server1 ~]# useradd -G desarrollo -c 'Grace Hopper' ghopper
```

```
[root@server1 ~]# useradd -G redes -c 'Hedy Lamarr' hlamarr
```

```
[root@server1 ~]# useradd -G redes -c 'Tim Berners' tberners
```

```
[root@server1 ~]# useradd -G redes -c 'Robert Kahn' rkahn
```

```
[root@server1 ~]# useradd -G seguridad -c 'Alan Turing' aturing
```

```
[root@server1 ~]# useradd -G seguridad -c 'Tatu Ylonen' tylonen
```

```
[root@server1 ~]# useradd -G sistemas -c 'Linus Torvalds' ltorvalds
```

```
[root@server1 ~]# useradd -G sistemas -c 'Dennis Ritchie' dritchie
```

Añadimos los directores al grupo directiva:

```
[root@server1 ~]# for dir in {alovelace,hlamarr,aturing,ltorvalds};do
```

```
    usermod -aG directiva $dir;done
```

```
[root@server1 ~]# tail -14 /etc/group | head -5
```

```
desarrollo:x:1101:alovelace,ghopper
```

```
redes:x:1102:hlamarr,tberners,rkahn
```

```
seguridad:x:1103:aturing,tylonen
```

```
sistemas:x:1104:ltorvalds,dritchie
```

```
directiva:x:1105:alovelace,hlamarr,aturing,ltorvalds
```

Les pongo a todos la contraseña *changeme* y obligo a que la cambien en el primer *login*:

```
[root@server1 ~]# for dpto in {desarrollo,redes,seguridad,sistemas,directiva}; do
```

```
for user in $(groupmems -g $dpto -l);do echo changeme | passwd --stdin $user;
```

```
chage -d 0 $user;done;done
```

3.5.- Obteniendo acceso root

El usuario *root* es el usuario con privilegios totales en el sistema. Para realizar determinadas tareas, como instalar paquetes, gestionar usuarios, etc., un usuario necesita escalar privilegios.

Se puede obtener privilegios de root de dos formas, bien cambiando del usuario con el que hayamos hecho *login* al usuario *root* usando el comando *su* o bien ejecutando el comando privilegiado con *sudo*.

3.5.1.- Comando su

Permite a un usuario hacer *login* con otro usuario del mismo sistema. Si no se especifica usuario, se cambia al usuario root. Si se ejecuta con un usuario regular, preguntará la contraseña del usuario al que queremos pasar pero si se ejecuta como *root*, pasaremos a hacer *login* con ese usuario sin necesidad de meter la contraseña del nuevo usuario.

Hay dos formas de hacer *login* a otro usuario:

- *no-login shell*: usando el comando `su <usuario>`. Con esto en el nuevo usuario heredamos las variables de entorno y lo especificado en los archivos de inicialización del usuario de origen en lugar del usuario al que hemos cambiado.
- *login shell*: usando `su - <usuario>` o `su -l <usuario>` o `su -login <usuario>`. Así cargaremos todo el entorno del nuevo usuario, es decir, se cargan los archivos `.bashrc` y `.bash_profile` del usuario al que queremos cambiar. Esta opción es la recomendable.

3.5.2.- Comando sudo

Con el comando `sudo` permitimos ejecutar ciertas cosas con privilegios de otro usuario sin necesidad de conocer la contraseña de dicho usuario. Se suele usar para delegar tareas de administración del sistema sin proporcionar la contraseña de *root*. El comando `sudo` pide la contraseña del propio usuario.

`Sudo` proporciona un control muy exhaustivo de que puede hacer un usuario con permisos de otro además de registrar todo lo que hace un usuario usando `sudo` en el archivo `/var/log/secure`.

→ Uso de sudo

Para usar `sudo`, se ejecuta:

```
sudo -u <usuario> <comando>
```

donde `<usuario>` es el usuario con el que se quiere ejecutar el comando `<comando>`. Si no se pone usuario, se hará como usuario *root*.

`Sudo` con redirecciones y/o pipes no funciona de forma adecuada, en su lugar hay que ejecutar con `sudo` el comando **`bash -c`** y a continuación el comando/s con redirecciones y/o pipes. Por ejemplo, introducir una nueva línea en el archivo `/etc/hosts`:

```
sudo bash -c "echo '10.11.1.101 server1 server1.example.com' >> /etc/hosts"
```

Opciones interesantes del comando `sudo`:

- l: veo los comandos que puedo utilizar con `sudo` con el usuario con el que he hecho *login*.
- e `<archivo>`: me permite editar un archivo como si fuese *root*.
- V: veo las opciones por defecto establecidas para todos los usuarios, comandos, equipos, ...

→ **Configuración de sudo**

Para configurar sudo se modifica el archivo principal */etc/sudoers* o se añade un archivo con el mismo formato a */etc/sudoers.d/*. Estos archivos se deben editar con el comando *visudo* ejecutado como *root* que bloquea el archivo para que sólo haya un usuario modificándolo además de hacer una verificación de la sintaxis del archivo antes de guardar los cambios. Si hubiese errores de sintaxis, tenemos tres opciones:

- e: editar de nuevo el archivo
- x: salir sin guardar los cambios
- Q: guardar los cambios y salir.

Para verificar la sintaxis de *sudoers* sin hacer cambios, basta con hacer *visudo -c* para que chequee */etc/sudoers* y *visudo -c -f <archivo>* en el caso de otro archivo.

Existe un grupo predefinido llamado *wheel* cuyos usuarios pueden usar sudo para ejecutar cualquier comando como si fuesen root. En CentOS 7/RHEL 7 la configuración para que los miembros del grupo *wheel* puedan ejecutar comandos privilegiados en el */etc/sudoers* ya viene por defecto activada, no así en las versiones anteriores que habría que descomentar la línea:

```
# %wheel ALL=(ALL) NOPASSWD: ALL
```

El archivo *sudoers* está basado en la forma EBNF que describe la gramática de un lenguaje que se va creando a través de reglas de producción que a la vez son la base para ser referenciadas por otras reglas.

Existen tres secciones, ninguna obligatoria, en el archivo */etc/sudoers*:

Alias: engloban bajo un nombre varios usuarios, equipos o comandos. Cada línea tiene el formato

```
<tipo_alias> <NOMBRE_ALIAS> = elemento1, elemento2, elemento3, ... elementoN
```

donde:

<tipo_alias> es *Cmnd_Alias* (alias de comandos), *User_Alias* (alias de usuarios normales), *Runas_Alias* (alias de usuarios administradores o con privilegios) y *Host_Alias* (alias de hosts).

<NOMBRE_ALIAS>: formado por caracteres alfanuméricos o *_* y debe comenzar por mayúscula. Se suelen poner en mayúsculas.

<elemento1>,: dependiendo del tipo de alias que sea serán comandos, *usernames*, %gruponames, *hostnames*, IPs, ...

Opciones o *defaults*: permiten definir ciertas características de comportamiento para los alias previamente creados, para usuarios, usuarios privilegiados, para equipos o de manera global para todos.

Reglas de acceso: definen qué usuarios ejecutan qué comandos bajo qué usuario y en qué equipos. Se configura la regla en una línea con el formato:

`<usuario> <host> = (<usuario_ejecucion>:<grupo_ejecucion>)<comando>`

donde:

`<usuario>`: puede ser un *username* o un alias de usuario. También puede aparecer un *groupname* anteponiendo el carácter `!`.

`<host>` puede ser ALL (cualquier equipo), un hostname, un alias de equipo, una dirección IP o una definición de red IP/máscara.

`<usuario_ejecución>` y `<grupo_ejecución>`: usuario y/o grupo con el que se ejecutará el comando. Los grupos se indican poniendo el carácter `%` delante del *groupname*. Si no aparece, el usuario/grupo de ejecución será *root*.

`<comando>`,...: cualquier comando dando su ruta completa. Si se termina en `'/'` indica todos los archivos dentro de ese directorio. Se pueden poner varios comandos separados por comas. Si el comando aparece entre comillas dobles, quiere decir que se permite el uso del comando pero sin parámetros ni argumentos.

A los comandos se puede anteponer una etiqueta que modifica como se ejecuta el comando. La etiqueta se hereda en una lista de comandos. Las posibles etiquetas son:

- ***NOPASSWD*** y ***PASSWD***: no requiere contraseña y requiere contraseña respectivamente. Por defecto si no aparece es *PASSWD*.
- ***NOEXEC*** y ***EXEC***: importante cuando el comando permite escapes a shell, p.e. `vi` usando `!`. Con *NOEXEC* no permitimos que esto suceda.
- ***SETENV*** y ***NOSETENV***: permite al usuario cambiar el entorno de variables a las del usuario con el que se ejecutará el comando.

→ **Ejemplos de reglas**:

admin ALL=(ALL:ALL) ALL

El usuario con *username* *admin* en cualquier *host*, puede ejecutar cualquier comando de cualquier usuario incluyendo a *root*.

fperez ALL = /sbin/iptables

El usuario con username *fperez* en cualquier *host* puede utilizar *iptables*.

ADMIN ALL = ALL

Los usuarios definidos en el alias *ADMIN* desde cualquier *host* pueden ejecutar cualquier comando.

%developers dbserver = (jboss) /opt/jboss/bin/standalone, (root) /var/log/*

Los usuarios que pertenezcan al grupo con *groupname developers* pueden en el sistema con *hostname dbserver* ejecutar como si fueran el usuario *jboss* el comando *standalone* y además como usuario *root* pueden ver los archivos del directorio */var/log*.

agarcia ALL = /usr/bin/passwd *, !/usr/bin/passwd root

El usuario con *username agarcia* desde cualquier sistema puede cambiar la contraseña a cualquier usuario excepto al usuario *root*.

flopez ALL = "/sbin/lsmmod"

El usuario con *username flopez* puede ejecutar el comando *lsmmod* pero sin argumentos luego sólo podrá listar los módulos del kernel.

pedro webserver = NOPASSWD: /bin/kill, /usr/bin/lprm, PASSWD:/etc/httpd/conf/, NOEXEC:/usr/bin/vi

El usuario con *username pedro* en el sistema con *hostname webserver* no necesita contraseña para los comandos *kill*, *lprm*, permite modificar todos los archivos que están en el directorio */etc/httpd/conf* necesitando contraseña y no permite ejecutar shell dentro del comando *vi*.

3.5.3.- Caso práctico

En **server1** vamos a permitir al usuario *admin* que pueda cambiar las password de cualquier usuario excepto *root*:

A los usuarios del grupo *redes*, les permitiremos usar los comandos de red *ip*, *ss*, *nmcli* y *firewall-cmd*, además de acceso a los archivos de red en */etc/sysconfig/network-scripts/*.

A los usuarios del grupo *sistemas*, permisos totales de *root* en el sistema.

RESOLUCIÓN

- Para permitir al usuario *admin* que pueda cambiar las contraseñas de todos los usuarios excepto *root*, en la parte de reglas del archivo */etc/sudoers* (editándolo con *visudo*) añadido la línea:

admin ALL = /usr/bin/passwd *, !/usr/bin/passwd root

- Para permitir a los usuarios del grupo *redes* que puedan ejecutar *ip*, *nmcli* y *firewall-cmd* y acceso al directorio, en la parte de alias del archivo *sudoers*, añadido la línea:

Cmdnd_Alias RED=/usr/sbin/ip,/usr/bin/nmcli,/usr/sbin/ss,/usr/bin/firewall-cmd, /etc/sysconfig/network-scripts/

y en la parte de reglas:

```
%sistemas ALL=RED
```

- Para dar acceso root a los usuarios del grupo *sistemas*, basta con añadirlos al grupo *wheel* y verificar que la configuración en el */etc/sudoers* para *wheel* está descomentada:

```
[root@server1 ~]# for user in $(groupmems -g sistemas -l);do usermod -aG wheel $user;done
```

```
[root@server1 ~]# less /etc/sudoers | grep wheel
```

```
## Allows people in group wheel to run all commands
```

```
%wheel    ALL=(ALL)  ALL
```

4. PERMISOS EN ARCHIVOS Y DIRECTORIOS

4.1.- Introducción

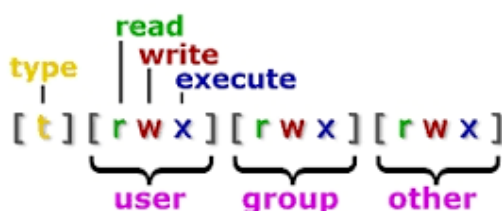
En Linux el acceso a archivos y directorios por parte de los usuarios se controla basándose en permisos.



Los permisos están formados por 9 letras: *r*, *w* y *x* que indican, en el caso de archivos, permiso de lectura, escritura y ejecución respectivamente. En el caso de directorios, *r* lectura indica que se puede listar el contenido del directorio, *w* escritura indica que se pueden crear o borrar archivos del directorio y *x* ejecución indica que el contenido del directorio puede ser accedido (informalmente llamado permiso de paso). En ambos casos, la ausencia de alguna de las tres letras, se representa por el carácter guión (-) e indica que no tiene ese permiso.

En los permisos hay tres bloques de tres letras cada uno, el primer bloque son los permisos del *usuario propietario* del archivo o directorio, el segundo bloque, *grupo propietario* y el tercer bloque, el resto de usuarios que no son ni usuario propietario ni grupo propietario, referido como *otros*.

La forma de leer los permisos es la siguiente: primero se mira si el usuario que quiere acceder al archivo o directorio es el usuario propietario, si lo es, le aplica el primer bloque de permisos. Si no lo es, se mira si el usuario pertenece al grupo propietario del archivo o directorio, si lo es, le aplica el segundo bloque de permisos y si ni es usuario propietario ni pertenece al grupo propietario, entonces le aplica el tercer bloque de permisos.



Tipos de permisos

Para ver los permisos de un archivo usaremos el comando `ls -l` y `ls -ld` en el caso de directorios.

4.2.- Gestionar permisos básicos

4.2.1.- Cambiar permisos de un archivo o directorio

El usuario propietario del archivo o directorio, o *root*, utilizará el comando `chmod` con la sintaxis:

```
chmod <permisos> <archivo>   o   chmod <permisos> <directorio>
```

en el caso de directorios se suele utilizar la opción `-R` cuando se quiere aplicar esos permisos de forma recursiva a todo el contenido del directorio.

Hay dos formas de expresar los permisos en *<permisos>*:

→ Método simbólico:

Se utiliza la sintaxis:

`chmod <quien><que><letra> <archivo> o chmod <quien><que><letra> <directorio>`

donde:

<quien>: es *u* para usuario, *g* para grupo, *o* para otros y *a* para todos.

<que>: es *+* para añadir, *-* para quitar y *=* para poner exactamente.

<letra>: es *r* para leer, *w* para escribir y *x* para ejecutar. En el caso de directorios, la *X* indica que sólo se da permiso de ejecución a los subdirectorios, no afecta a los archivos.

No es necesario dar todos los permisos, sólo los que se quieren modificar.

→ Método numérico:

Se utiliza la sintaxis:

`chmod ### <archivo> o chmod ### <directorio>`

donde *###* son tres (o cuatro dígitos en el caso de dar permisos especiales) de forma que el primer dígito es el permiso para usuario propietario, el segundo es el permiso para el grupo propietario y el tercer dígito es el permiso para otros.

Los dígitos van de 0 a 7, donde 0 indica que no tiene ese permiso y el dígito es el resultado de sumar los permisos, siendo *r=4*, *w=2* y *x=1*, de forma que si un archivo tiene permiso de lectura y escritura para usuario propietario, sólo lectura para grupo propietario y nada para otros, tendremos: $(4+2) = 6$ para usuario propietario, 4 para grupo propietario y 0 para otros quedando los tres dígitos 640.

En este método si es necesario dar todos los permisos, hay que poner los tres dígitos.

NOTA: cuando queramos cambiar el acceso al contenido de un archivo, se modifican los permisos del archivo, cuando queramos cambiar el que el archivo se pueda borrar, se modifican los permisos del directorio. Un usuario sin permisos en el archivo pero con permisos de escritura en el directorio que lo contiene, no puede acceder al contenido del archivo ¡pero sí puede borrarlo!.

4.2.2.- Cambiar el usuario y grupo propietario de un archivo o directorio

Se utiliza el comando *chown* para cambiar tanto usuario propietario como grupo propietario con la sintaxis:

`chown <usuario>:<grupo> <archivo> o chown <usuario>:<archivo> <directorio>`

Podemos sólo cambiar el usuario propietario, omitiendo la parte :<grupo> o sólo cambiar el grupo propietario, omitiendo <usuario> y dejando :<grupo> (importante dejar los :).

También se puede usar el comando *chgrp* para cambiar el grupo propietario con la sintaxis:

<grupo> <archivo> o *chgrp* <grupo> <directorio>

En ambos comandos, *chown* y *chgrp*, se puede usar la opción *-R* en el caso de directorios para aplicar el cambio de usuario y/o grupo de forma recursiva al contenido del directorio.

Hay que tener en cuenta que el usuario propietario de un archivo o directorio, sólo root puede cambiarlo. Un usuario regular sólo podrá cambiar el grupo propietario a sus archivos o directorios si pertenece al grupo al cual lo cambia.

4.3.- Permisos especiales

Tenemos 3 tipos de permisos especiales:

- *setuid en un archivo ejecutable*: cuando se ejecute el archivo lo hará como si lo ejecutase su usuario propietario y no el usuario que lo está ejecutando.
- *setgid en un archivo ejecutable*: cuando se ejecute el archivo lo hará como si lo ejecutase su grupo propietario y no el grupo del usuario que lo está ejecutando.
- *setgid en directorio*: los archivos creados en ese directorio, por defecto tendrán el grupo propietario igual al del directorio. Usado para crear directorios colaborativos.
- *sticky bit en directorio*: los usuarios que pueda escribir en el directorio, sólo podrán borrar los archivos que son suyos y no podrán borrar los de otros.

El *setuid* en directorio y el *stickybit* en archivo no tienen sentido, no causan ningún efecto si se activan.

Estos permisos especiales aparecen codificados en los tres bloques de los permisos normales, en el tercer dígito de cada bloque. El *setuid* aparece en el bloque del usuario propietario, en lugar de aparecer x, aparece s. El *setgid* aparece en el bloque de grupo propietario, en lugar de x, aparece s. Y el *stickbit* aparece en el bloque de otros, en lugar de x aparece una t.

Si aparecen en mayúsculas, S o T, significa que el permiso x correspondiente no existe, hay un -.

4.3.1.- Activar y desactivar permisos especiales

De forma simbólica:

- Para activar: *setuid*=u+s, *setgid*=g+s y *stickybit*=o+t.
- Para desactivar: *setuid*=u-s, *setgid*=g-s y *stickybit*=o-t.

De forma numérica, se pone un dígito delante de los tres dígitos de los permisos, *setuid*=4, *setgid*=2 y *stickybit*=1. Si aparece un 3 en un directorio, significa que tiene el *setgid* y el *stickybit* activados.

4.4.- Permisos por defecto

Los permisos por defecto de los archivos y directorios los configuran los procesos que los crean. Por defecto los directorios se crean con permisos 777 y los archivos con 666 pero existe lo que se llama la máscara, que borra el permiso que tenga activado. La máscara por defecto es 0002, que hace que los directorios cuando se creen, pasen a tener 775 en lugar de 777 y los archivos 664 en lugar de 666.

Para ver el valor de la máscara se ejecuta el comando *umask* sin argumentos. Para cambiar la máscara por defecto, se utiliza el comando *umask* seguido de 4 dígitos que serán la nueva máscara.

4.5.- ACLs

Hay ocasiones en que se necesita un control más fino sobre los permisos en los archivos y directorios, por ejemplo, permitir a un usuario en concreto acceder a determinado archivo sin que sea ni usuario propietario ni pertenezca al grupo propietario del archivo.

Las ACLs (Access Control List) aplican a usuarios y grupos específicamente nombrados en la ACL y al grupo propietario, no afectan en ningún caso al propietario del archivo o directorio o a otros.

Los nuevos archivos y subdirectorios que se crean, heredan las ACLs por defecto que tenga configuradas en directorio que los contiene.

El sistema de archivos que contiene los archivos y directorios con ACLs tiene que soportarlo. En el caso de XFS, el sistema de archivos por defecto en CentOS, ya lo soporta y lo activa por defecto. Los sistemas de archivos de tipo EXT4 también pero no así versiones anteriores como EXT3 y EXT2.

4.5.1.- Ver e interpretar ACLs

Cuando ejecutamos *ls -l* sobre un archivo o *ls -ld* sobre un directorio, podemos ver si tiene una ACL configurada en el último carácter tras los permisos. Si es un carácter + tiene ACL configurada, si aparece un punto, no la tiene.

Si hay ACL configurada, los permisos de usuario propietario y otros no cambian pero los permisos que aparecen en el bloque de grupo propietario, ya no son esos permisos, lo que aparece corresponde a la máscara de la ACL. Para ver los permisos del grupo propietario tendremos que ver en detalle la ACL.

Para ver en detalle la ACL, tendremos que ejecutar el comando *getfacl* sobre el archivo directorio.

La información que aparece es, primero 3 líneas comentadas con el nombre del archivo o directorio, usuario propietario y grupo propietario; seguido de la línea con los permisos del usuario propietario. A continuación aparecen las líneas con los permisos para los usuarios con un nombre o *uid* en

concreto, luego los permisos del grupo propietario, indicado con ::, y luego los grupos con nombre o gid en concreto. Por último aparece la máscara y los permisos de otros.

En el caso de un directorio, además al final aparece la configuración de la ACL por defecto, la ACL que heredarán todos los archivos y subdirectorios que se creen dentro del directorio.

La máscara marca los permisos máximos que cualquier usuario afectado por la ACL podrá tener.

La precedencia en los permisos cuando un usuario accede a un archivo o directorio con ACL configurada es:

- Si el usuario es usuario propietario, le aplican los permisos normales, el primer bloque en los permisos del archivo o directorio.
- Si el usuario es un usuario con nombre, de los que salen listados en la ACL, le aplican los permisos que ahí tenga como *effective*.
- Si el usuario pertenece al grupo propietario, le aplican los permisos que aparecen en la ACL para el grupo propietario como *effective*.
- Si el usuario pertenece a algún grupo con nombre, los que aparecen listados en la ACL, le aplican los permisos que ahí tengan como *effective*.
- Si no ha cumplido ninguna condición anterior, le aplican los permisos normales de otros, los del tercer bloque de permisos del archivo o directorio.

4.5.2.- Configurar ACLs

La ACL utiliza la representación normal de los permisos con las letras *r* para lectura, *w* para escritura y *x* para ejecución. Un guión indica ausencia de ese permiso y la *X* indica que el permiso de ejecución sólo se activará para directorios.

Se utiliza el comando *setfacl* con la opción *-m* para modificar seguido de:

u:<username>:<permisos> <archivo>: se añaden esos permisos para el usuario dado.

g:<groupname>:<permisos> <archivo>: se añaden esos permisos para el grupo dado.

g::<permisos> <archivo>: se añaden esos permisos para el grupo propietario.

En lugar de nombre del usuario se puede poner el *uid* y en lugar del nombre del grupo se puede poner el *gid*. Para directorios sería igual utilizando la opción *-R* para que lo haga recursivo.

Si en un archivo con ACL configurada, hacemos *chmod g+|-<permisos> <archivo>* no estamos cambiando los permisos del grupo propietario, estamos cambiando la máscara de la ACL.

Hacer *setfacl -m u::<permisos> <archivo>* es lo mismo que *chmod u+|-<permisos> <archivo>* y hacer *setfacl -m o::<permisos> <archivo>* es lo mismo que *chmod o+|-<permisos> <archivo>*; no hay diferencia.

4.5.3.- Eliminar ACLs

Para eliminar una entrada específica en una ACL configurada, se utiliza la opción `-x` del comando `setfacl` con lo que se quiera eliminar.

Para eliminar una entrada por defecto específica en una ACL de un directorio, se utiliza la opción `-x` y delante de lo que se quiere eliminar, hay que poner tal y como viene en la ACL, `d:`.

Para eliminar las ACL por defecto de un directorio, se utiliza la opción `-k`.

Para borrar toda la ACL, incluidas las de por defecto de un directorio, se utiliza la opción `-b` con el nombre del archivo o directorio a continuación.

4.6.- Caso práctico

En **server1**, para los grupos de los departamentos creados anteriormente: *desarrollo*, *sistemas*, *redes*, *directiva* y *seguridad*, se van a crear directorios colaborativos en `/` con el nombre del departamento.

Además para el departamento de *desarrollo*, *directiva* y *redes*, sólo el usuario que cree un archivo, podrá eliminarlo.

Al usuario *admin* se le creará una ACL para que pueda acceder al contenido del directorio de sistemas.

RESOLUCIÓN

- Creamos los directorios para cada departamento:

```
[root@server1 ~]# for depto in {desarrollo,sistemas,redes,seguridad,directiva};do
                                                                mkdir /$i;done
```

- Los hacemos colaborativos, debe pertenecer al grupo, tener permiso de escritura para el grupo y tener el *setgid* activado y en el caso de *desarrollo*, *directiva* y *redes*, además activar el *stickybit*:

```
[root@server1 ~]# chgrp desarrollo /desarrollo; chmod g+ws,o+t /desarrollo
```

```
[root@server1 ~]# chgrp directiva /directiva; chmod g+ws,o+t /directiva
```

```
[root@server1 ~]# chgrp redes /redes; chmod g+ws,o+t /redes
```

```
[root@server1 ~]# chgrp seguridad /seguridad; chmod g+ws /seguridad
```

```
[root@server1 ~]# chgrp sistemas /sistemas; chmod g+ws /sistemas
```

- Verificamos que usando p.e. el usuario *ghopper*, cuando crea un archivo en el directorio de su departamento */desarrollo*, aparece como grupo propietario el del directorio:

```
[root@server1 ~]# su - ghopper
```

```
[ghopper@server1 ~]$ touch /desarrollo/hola.txt
```

```
[ghopper@server1 ~]$ ls -ld /desarrollo
```

```
drwxrwsr-t. 2 root desarrollo 6 Sep 18 13:10 /desarrollo/
```

```
[ghopper@server1 ~]$ ls -l /desarrollo
```

```
total 0
```

```
-rw-rw-r--. 1 ghopper desarrollo 0 Sep 18 13:13 hola.txt
```

- Creamos la ACL y ACL por defecto para el usuario *admin* en */sistemas*:

```
[root@server1 ~]# setfacl -Rm u:admin:rwX /sistemas
```

```
[root@server1 ~]# setfacl -m d:u:admin:rwX /sistemas
```

```
[root@server1 ~]# getfacl /sistemas
```

```
getfacl: Removing leading '/' from absolute path names
```

```
# file: sistemas
```

```
# owner: root
```

```
# group: sistemas
```

```
# flags: -s-
```

```
user::rwX
```

```
user:admin:rwX
```

```
group::rwX
```

```
mask::rwX
```

```
other::r-x
```

```
default:user::rwX
```

```
default:user:admin:rwX
```

```
default:group::rwX
```

```
default:mask::rwX
```

```
default:other::r-x
```

- Comprobamos que *admin* puede crear un archivo en el directorio */sistemas*:

```
[admin@server1 ~]$ touch /sistemas/info.txt
```

```
[admin@server1 ~]$ ls -l /sistemas/
```

```
total 0
```

```
-rw-rw-r--+ 1 admin sistemas 0 Sep 18 13:19 info.txt
```