PRÁCTICA 1: CONTROL DE ACCESOS EN UNIX

EDUARDO MORA GONZÁLEZ

Ejercicio 1: Consultar los manuales de passwd.

A) Averiguar las restricciones que se imponen al formato del password.

Las restricciones que el manual de *passwd* propone son las siguientes:

- Debe tener entre 6 y 8 caracteres, ya que es el formato estándar.
- Usar el alfabeto en minúsculas.
- Usar dígitos del 0 al 9.
- Usar signos de puntuación.

No obstante, *passwd* puede rechazar la contraseña si no es lo suficientemente compleja.

B) Resumir los plazos y medidas de envejecimiento aplicables a los passwords en el sistema linux.

El tiempo mínimo que se puede tener un password se establece con **-n**, si es 0 el password se puede cambiar cuando se quiera.

El tiempo máximo que se puede tener un password se establece con **-x**, a partir de ese momento se requiere que se cambie el password, y esto se puede hacer durante un tiempo establecido con **-w**, si ese tiempo pasa, el pasword del usuario expira.

Ejercicio 2: Se puede encontrar una descripción de las principales tareas de administración. Entre ellas se pueden encontrar algunas medidas relacionadas con la seguridad del sistema. Resumir las fundamentales.

Las medidas fundamentales son:

- Poner en marcha nuevos servidores.
- Cambiar la configuración hardware de los equipos.
- Instalar nuevo software y actualizar el existente.
- Gestión de cuentas de usuarios.
- Monitorizar el rendimiento del sistema.
- Atención usuarios.
- Reporte a dirección/organización.
- Documentación del sistema.
- Realizar backups: Una política de backups con frecuencias y niveles que salvaguarde adecuadamente la información ante posibles fallos.
- Fallos y caídas del sistema: Si no supimos detectarlo, o si el fallo se produce súbitamente, es necesaria una intervención rápida para restaurar el sistema a su funcionamiento normal.

- Seguridad: No hay ningún sistema infalible ante un acceso no autorizado o un ataque, pero está en su mano tener las políticas adecuadas en marcha.

Ejercicio 3: Consultar el fichero /etc/passwd y comprobar los usuarios especiales existentes y los números de usuario utilizados.

La salida del fichero es la siguiente:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
uuidd:x:105:110::/run/uuidd:/usr/sbin/nologin
avahi-autoipd:x:106:111:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:107:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:108:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
rtkit:x:109:114:RealtimeKit,,,:/proc:/usr/sbin/nologin
cups-pk-helper:x:110:116:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
whoopsie:x:111:117::/nonexistent:/bin/false
kernoops:x:112:65534:Kernel Oops Tracking Daemon,,,:/:/usr/sbin/nologin
saned:x:113:119::/var/lib/saned:/usr/sbin/nologin
pulse:x:114:120:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
avahi:x:115:122:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
colord:x:116:123:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
hplip:x:117:7:HPLIP system user,,,:/var/run/hplip:/bin/false
mpd:x:118:29::/var/lib/mpd:/usr/sbin/nologin
sddm:x:119:124:Simple Desktop Display Manager:/var/lib/sddm:/bin/false
geoclue:x:120:125::/var/lib/geoclue:/usr/sbin/nologin
emorag:x:1000:1000:emorag,,,:/home/emorag:/bin/bash
```

Los usuarios especiales son incorporados por el propio sistema, y se encargan de administrar los demonios del mismo. Estos usuarios no pueden iniciar sesión en el sistema ni tener un shell donde trabajar, por tanto, no tienen contraseña asignada.

Estos usuarios se identifican porque su UID es un numero bajo (que se encuentra entre el 0 y el 999), si estos UID's se acaban se van añadiendo más con números más elevados.

Ejercicio 4. Crear un guion del Shell que dado un nombre de usuario (login) obtenga su UID y GID.

El programa Shell llamada Ejercicio_4.sh es el siguiente:

```
#!/bin/bash

USUARIO=$1

id $USUARIO 1>/dev/null 2>&1

ERR=$?

if [ $ERR -ne 0 ]
then
    echo EL USUARIO $USUARIO no existe
    exit
fi

NOMBRE=$(id $USUARIO | cut -f1 -d "")

echo UID: $NOMBRE
```

Después de darle los permisos convenientes de ejecución, la salida es la siguiente:

```
emorag@emorag:~/Escritorio/PRACTICA$ ./Ejercicio_4.sh
UID: uid=1000(emorag) gid=1000(emorag) grupos=1000(emorag),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),116(lpadmin),126(sambashare)
emorag@emorag:~/Escritorio/PRACTICA$
```

Ejercicio 5. Crear un guion del Shell que dado un GID liste los logins de todos los usuarios que lo componen.

El programa Shell llamada Ejercicio_5.sh es el siguiente:

```
#!/bin/bash

awk -F ':' '{print $1}' /etc/passwd
```

Después de darle los permisos convenientes de ejecución, la salida es la siguiente:

```
emorag@emorag:~/Escritorio/PRACTICA$ ./Ejercicio_5.sh
root
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
systemd-network
systemd-resolve
syslog
messagebus
_apt
uuidd
avahi-autoipd
usbmux
dnsmasq
rtkit
cups-pk-helper
whoopsie
kernoops
saned
pulse
avahi
colord
hplip
mpd
sddm
geoclue
emorag
emorag@emorag:~/Escritorio/PRACTICA$
```

Ejercicio 6. Consultar el manual de la orden pwck y practicar su uso.

El comando **pwck** verifica la integridad de los usuarios e información de autenticación. Comprueba que todas las entradas en **/etc/passwd** y **/etc/shadow** tienen el formato adecuado y contienen datos válidos. Se realizan verificaciones para verificar que cada entrada tiene:

- El número correcto de campos.
- Un nombre de usuario único y válido.
- Un usuario válido y un identificador de grupo.
- Un grupo primario válido.
- Un directorio de inicio válido.
- Un Shell de inicio de sesión válido.

Para probar su uso se debe ejecutar en modo administrador tal y como se ve a continuación:

```
emorag@emorag:~/Escritorio/PRACTICA$ man pwck
emorag@emorag:~/Escritorio/PRACTICA$ pwck
pwck: Permission denied.
pwck: no se pudo bloquear /etc/passwd, inténtelo de nuevo.
emorag@emorag:~/Escritorio/PRACTICA$ sudo pwck
[sudo] contraseña para emorag:
usuario «lp»: el directorio «/var/spool/lpd» no existe
usuario «news»: el directorio «/var/spool/news» no existe
usuario «uucp»: el directorio «/var/spool/uucp» no existe
usuario «www-data»: el directorio «/var/www» no existe
usuario «list»: el directorio «/var/list» no existe
usuario «irc»: el directorio «/var/run/ircd» no existe
usuario «gnats»: el directorio «/var/lib/gnats» no existe
usuario «nobody»: el directorio «/nonexistent» no existe
usuario «syslog»: el directorio «/home/syslog» no existe
usuario «messagebus»: el directorio «/nonexistent» no existe
usuario «_apt»: el directorio «/nonexistent» no existe
usuario «usbmux»: el directorio «/var/lib/usbmux» no existe
usuario «cups-pk-helper»: el directorio «/home/cups-pk-helper» no existe
usuario «whoopsie»: el directorio «/nonexistent» no existe
usuario «saned»: el directorio «/var/lib/saned» no existe
usuario «pulse»: el directorio «/var/run/pulse» no existe
usuario «hplip»: el directorio «/var/run/hplip» no existe
pwck: sin cambios
emorag@emorag:~/Escritorio/PRACTICA$
```

Ejercicio 7. Practicar el uso la orden chmod en modo simbólico realizando los siguientes ejercicios:

A) Crear un fichero con permisos r-x------
B) Dar permiso de ejecución a los usuarios del grupo
C) Dar permiso de lectura a todos los usuarios
D) Modificar los permisos del grupo para que sean rwx
E) Hacer que los permisos del fichero sean rwxr--r—

```
emorag@emorag:~/Escritorio/PRACTICA$ gvim EJERCICIO_7.txt
emorag@emorag:~/Escritorio/PRACTICA$ chmod 500 EJERCICIO_7.txt
emorag@emorag:~/Escritorio/PRACTICA$ chmod g+x EJERCICIO_7.txt
emorag@emorag:~/Escritorio/PRACTICA$ chmod a+r EJERCICIO_7.txt
emorag@emorag:~/Escritorio/PRACTICA$ chmod g+rwx EJERCICIO_7.txt
emorag@emorag:~/Escritorio/PRACTICA$ chmod 776 EJERCICIO_7.txt
```

Ejercicio 8. Crear un directorio con todos los permisos para el propietario.

A) Situarse dentro y crear un fichero cuya ejecución diga "Hola Mundo" o cualquier frase de tu interés. Hacer ocho copias de ese fichero y darle a cada una de las 8 combinaciones de permisos para el propietario (0 a 7).

```
emorag@emorag:~/Escritorio/PRACTICA$ mkdir EJERCICIO_8
emorag@emorag:~/Escritorio/PRACTICA$ chmod 700 EJERCICIO_8
emorag@emorag:~/Escritorio/PRACTICA$ gvim fichero_1.sh
```

```
emorag@emorag:~/Escritorio/PRACTICA$ gvim fichero_1.sh
emorag@emorag:~/Escritorio/PRACTICA$ cp fichero_1.sh fichero_2.sh
emorag@emorag:~/Escritorio/PRACTICA$ cp fichero_1.sh fichero_3.sh
emorag@emorag:~/Escritorio/PRACTICA$ cp fichero_1.sh fichero_4.sh
emorag@emorag:~/Escritorio/PRACTICA$ cp fichero_1.sh fichero_5.sh
emorag@emorag:~/Escritorio/PRACTICA$ cp fichero_1.sh fichero_6.sh
emorag@emorag:~/Escritorio/PRACTICA$ cp fichero_1.sh fichero_7.sh
emorag@emorag:~/Escritorio/PRACTICA$
emorag@emorag:~/Escritorio/PRACTICA$
emorag@emorag:~/Escritorio/PRACTICA$ chmod 000 fichero_1.sh
emorag@emorag:~/Escritorio/PRACTICA$ chmod 100 fichero_2.sh
emorag@emorag:~/Escritorio/PRACTICA$ chmod 300 fichero_3.sh
emorag@emorag:~/Escritorio/PRACTICA$ chmod 400 fichero_4.sh
emorag@emorag:~/Escritorio/PRACTICA$ chmod 500 fichero_5.sh
emorag@emorag:~/Escritorio/PRACTICA$ chmod 600 fichero_6.sh
emorag@emorag:~/Escritorio/PRACTICA$ chmod 700 fichero_7.sh
```

B) Intentar deducir el funcionamiento de las siguientes ordenes con cada uno de los ficheros: cat, cp, vi(lectura), vi(escritura), rm, ejecución.

Viendo los permisos asignados a cada uno de los ficheros, se puede deducir que:

- Sólo se puede ejecutar el comando **cat** sobre aquellos ficheros que tengan permiso de lectura, en los ficheros creados son el número 4, 5, 6 y 7.
- Sólo se puede ejecutar el comando **cp** sobre aquellos ficheros que tengan permiso de lectura, en los ficheros creados son el número 4, 5, 6 y 7.
- Sólo se puede ejecutar el comando **vi (lectura)** sobre los ficheros que tengan permiso de lectura, en los ficheros creados son el número 4, 5, 6 y 7.
- Sólo se puede ejecutar el comando **vi (escritura)** sobre aquellos ficheros que tengan permiso de lectura y escritura, en los ficheros creados son el número 2, 3, 6 y 7.
- Podremos ejecutar **rm** sobre todos los archivos.
- Sólo podremos **ejecutar** aquellos archivos que tengan permiso de ejecución, en los ficheros creados son el número 1, 3, 5 y 7.

Solo el archivo que tiene los permisos 000 se nos negarán todos los comandos menos el de borrar.

C) Comprobar el funcionamiento de cada una de las ordenes anteriores.

	cat	cp	Vi (lectura)	Vi (escritura)	rm	ejecutar
Fichero_0	NO	NO	NO	NO	SI	NO
Fichero_1	NO	NO	NO	NO	SI	NO
Fichero_2	NO	NO	NO	NO	SI	NO
Fichero_3	NO	NO	NO	NO	SI	NO
Fichero_4	SI	SI	SI	NO	SI	NO
Fichero_5	SI	SI	SI	NO	SI	SI
Fichero_6	SI	SI	SI	SI	SI	NO
Fichero_7	SI	SI	SI	SI	SI	SI

Ejercicio 9. Modifica los permisos del directorio creado en el ejercicio anterior eligiendo una de las 8 combinaciones de permisos posibles.

```
emorag@emorag:~/Escritorio/PRACTICA$ chmod 600 EJERCICIO_8/
emorag@emorag:~/Escritorio/PRACTICA$
```

A) Intentar deducir el funcionamiento de cada una de las siguientes ordenes en base al significado de los permisos sobre directorios:

- **Ordenes de listado: ls, ll** → Se ejecutan ambos comandos.
- **Orden de acceso: cd** → No se permite el acceso.
- **Ordenes de modificación del directorio: cp (hacia el directorio)** → No se permite copiar ya que no se permite acceder a él.
- **rm (fichero del directorio)** → Si se ejecuta el comando.
- **Otras ordenes: cp (desde el directorio)** → No se permite al no poder entrar en él.
- **cat (un fichero del directorio)** → Si se pudiera entrar si se permitiera, pero al no poder entrar no se puede ejecutar el comando.
- **¿Qué ordenes dependerán de los permisos de los ficheros afectados, además de los permisos del directorio que los contiene?** → Todas las ordenes menos la de borrado.

B) Comprobar el funcionamiento de las ordenes anteriores y resumirlo en una tabla.

	FUNCIONAMIENTO
LS	Muestra un listado con los archivos y directorios de un determinado directorio.
LL	Alias del comando LS (LL = LS -ALL) que muestra toda la información de los archivos y directorios.
CD	Sirve para mostrar el nombre del directorio actual y, también, permite cambiar de directorio.
CP	Permite copiar archivos, directorios, permisos, propiedades, enlaces …
RM	Permite eliminar archivos y directorios del sistema de archivos.
CAT	Permite leer datos de archivos y mostrar sus contenidos.

Ejercicio 10. ¿Cómo puedes conocer el modo de creación por defecto de un fichero?

Para saber el modo de creación de un fichero se debe usar el comando **umask**

Ejercicio 11. Si el modo de creación es 0666, ¿Qué efecto tienen las máscaras 0077, 0022 y 0002?

	Octal	Binario			
	0666	000	110	110	110
Máscara	0077	000	000	111	111
Complemento	(0077)	111	111	000	000
	0600	0	110	0	0
	0666	000	110	110	110
Máscara	0022	000	000	010	010
Complemento	(0022)	111	111	101	101
	0644	000	110	100	100
	0666	000	110	110	110
Máscara	0002	000	000	000	001
Complemento	(002)	111	111	111	110
	0664	000	110	110	110

¿Y si el modo de creación es 0644?

	Octal	Binario			
	0644	000	110	100	100
Máscara	0077	000	000	111	111
Complemento	(0077)	111	111	000	000
	600	000	110	000	000
	0644	000	110	100	100
Máscara	0022	000	000	010	010
Complemento	(0022)	111	111	101	101
	644	000	110	100	100
	0644	000	110	100	100
Máscara	0002	000	000	000	001
Complemento	(002)	111	111	111	110
	644	000	110	100	100

Ejercicio 12. ¿Cuál sería la máscara que crearía los ficheros sin permiso de escritura ni ejecución para los usuarios no propietarios de los mismos?

El permiso que necesitamos para crear los ficheros sin permiso de escritura ni ejecución para los usuarios no propietarios de los mismos sería el 0X44, por lo que se deduce que la máscara debería ser 0033.

Ejercicio 13. Escribir un guion con dos parámetros. Si el primer parámetro es -u se listarán los ficheros SUID y si es -g si listarán los SGID. El segundo parámetro será el nombre de un directorio en el que se buscarán ficheros SUID o SGID en función del primer parámetro.

```
1  #!/bin/bash
2
3  case $1 in
4
5          -u) find $2 -perm -4000 -ls ;;
6
7          -g) find $2 -perm -2000 -ls ;;
8
9  esac
```

Probar el guion sobre los directorios de tu path de búsqueda.

```
emorag@emorag:~/Escritorio/PRACTICA$ ./Ejercicio_13.sh -u /bin
  4596608     64 -rwsr-xr-x   1 root     root        64424 jun 28  2019 /bin/ping
  4596549     44 -rwsr-xr-x   1 root     root        43088 ago 23  2019 /bin/mount
  4596570     28 -rwsr-xr-x   1 root     root        26696 ago 23  2019 /bin/umount
  4596636     44 -rwsr-xr-x   1 root     root        44664 mar 22  2019 /bin/su
  4596535     32 -rwsr-xr-x   1 root     root        30800 ago 11  2016 /bin/fusermount
emorag@emorag:~/Escritorio/PRACTICA$
emorag@emorag:~/Escritorio/PRACTICA$ ./Ejercicio_13.sh -g /usr/bin
   920134     44 -rwxr-sr-x   1 root     mlocate     43088 mar  1  2018 /usr/bin/mlocate
   919531     40 -rwxr-sr-x   1 root     crontab     39352 nov 16  2017 /usr/bin/crontab
   919617     24 -rwxr-sr-x   1 root     shadow      22808 mar 22  2019 /usr/bin/expiry
   920714    356 -rwxr-sr-x   1 root     ssh        362640 mar  4  2019 /usr/bin/ssh-agent
   919485     72 -rwxr-sr-x   1 root     shadow      71816 mar 22  2019 /usr/bin/chage
   929955     32 -rwxr-sr-x   1 root     tty         30800 ago 23  2019 /usr/bin/wall
   919456     16 -rwxr-sr-x   1 root     tty         14328 ene 17  2018 /usr/bin/bsd-write
emorag@emorag:~/Escritorio/PRACTICA$
```

Ejercicio 14. Ejercicio de manejo de programas SUID.

A) Estudia el objetivo de las llamadas al sistema getuid() y geteuid() y system().

- **Getuid()** → Devuelve la ID de usuario real del proceso de llamada.
- **Geteuid()** → Devuelve la identificación de usuario efectiva del proceso de llamada.
- **System()** → Ejecuta un comando de Shell.

B) Crea un directorio llamado solomio con permisos 700. Dentro del mismo crea los ficheros fich1 y fich2.

```
emorag@emorag:~/Escritorio/PRACTICA$ mkdir solomio
emorag@emorag:~/Escritorio/PRACTICA$ chmod 700 solomio
emorag@emorag:~/Escritorio/PRACTICA$ cd solomio/
emorag@emorag:~/Escritorio/PRACTICA/solomio$ touch fich1
emorag@emorag:~/Escritorio/PRACTICA/solomio$ touch fich2
emorag@emorag:~/Escritorio/PRACTICA/solomio$
```

C) Copia el siguiente código:

```
1  #include <stdlib.h>
2  #include <stdio.h>
3
4  int main(){
5
6  printf("Real: %d, Efectivo: %d\n", getuid(), geteuid());
7  system("whoami");
8  system("pwd;ls -l");
9  system("ls -l solomio");
```

Denomina al fichero mio.c. Compílalo mediante la orden make para obtener el ejecutable mio. Ejecútalo e interpreta el resultado.

```
emorag@emorag:~/Escritorio/PRACTICA/solomio$ ls
a.out  fich1  fich2  mio.c
emorag@emorag:~/Escritorio/PRACTICA/solomio$ ./a.out
Real: 1000, Efectivo: 1000
emorag
/home/emorag/Escritorio/PRACTICA/solomio
total 16
-rwxrwxr-x 1 emorag emorag 8432 abr 14 13:28 a.out
-rw-rw-r-- 1 emorag emorag    0 abr 14 13:20 fich1
-rw-rw-r-- 1 emorag emorag    0 abr 14 13:20 fich2
-rw-rw-r-- 1 emorag emorag  216 abr 14 13:29 mio.c
ls: no se puede acceder a 'solomio': No existe el archivo o el directorio
emorag@emorag:~/Escritorio/PRACTICA/solomio$
```

El usuario efectivo y real son el mismo: alumno, el programa realiza un ls sobre la carpeta en la que estamos después de mostrarla con el pwd. No realiza el ls sobre solomio ya que ese directorio no se encuentra en la carpeta.

D) Copia en tu directorio el fichero del apartado b como fichero otroyo propiedad del usuario hacker. Ejecútalo e interpreta los resultados

```
emorag@emorag:~/Escritorio/PRACTICA/solomio$ sudo useradd hacker
useradd: el usuario «hacker» ya existe
emorag@emorag:~/Escritorio/PRACTICA/solomio$ cp mio.c otroyo.c
emorag@emorag:~/Escritorio/PRACTICA/solomio$ sudo chown hacker otroyo
```

```
emorag@emorag:~/Escritorio/PRACTICA/solomio$ ls
a.out  fich1  fich2  mio.c  otroyo.c
emorag@emorag:~/Escritorio/PRACTICA/solomio$ ./a.out
Real: 1000, Efectivo: 1000
emorag
/home/emorag/Escritorio/PRACTICA/solomio
total 20
-rwxrwxr-x 1 emorag emorag 8440 abr 14 13:54 a.out
-rw-rw-r-- 1 emorag emorag    0 abr 14 13:20 fich1
-rw-rw-r-- 1 emorag emorag    0 abr 14 13:20 fich2
-rw-rw-r-- 1 emorag emorag  216 abr 14 13:42 mio.c
-rw-rw-r-- 1 hacker emorag  216 abr 14 13:54 otroyo.c
ls: no se puede acceder a 'solomio': No existe el archivo o el directorio
```

En este caso el usuario efectivo y real del archivo otroyo es distinto.

E) También en tu directorio, copia el ejecutable suid, propiedad de hacker y SUID desde su ubicación original. Ejecútalo e interpreta los resultados.

```
emorag@emorag:~/Escritorio/PRACTICA$ cp otroyo mio.c
emorag@emorag:~/Escritorio/PRACTICA$ chmod u+s mio.c
```

```
emorag@emorag:~/Escritorio/PRACTICA$ ./a.out
Real: 1000, Efectivo: 1000
emorag
/home/emorag/Escritorio/PRACTICA
total 76
-rwxrwxr-x 1 emorag emorag 8432 abr 14 13:43  a.out
-rwxrwxr-x 1 emorag emorag  101 abr 10 20:57  Ejercicio_13.sh
-rw-rw-r-- 1 emorag emorag 2307 abr  7 08:36 'EJERCICIO 3'
-rwxrwxr-x 1 emorag emorag  190 abr  7 08:48  Ejercicio_4.sh
-rwxrwxr-x 1 emorag emorag   49 abr  7 08:51  Ejercicio_5.sh
-rwxrwxrw- 1 emorag emorag    5 abr  7 09:02  EJERCICIO_7.txt
drw------- 2 emorag emorag 4096 abr 10 20:13  EJERCICIO_8
---------- 1 emorag emorag    7 abr 10 19:40  fichero_1.sh
---x------ 1 emorag emorag    7 abr 10 19:40  fichero_2.sh
--wx------ 1 emorag emorag    7 abr 10 19:41  fichero_3.sh
-r-------- 1 emorag emorag    7 abr 10 19:41  fichero_4.sh
-r-x------ 1 emorag emorag    7 abr 10 19:41  fichero_5.sh
-rw------- 1 emorag emorag    7 abr 10 19:41  fichero_6.sh
-rwx------ 1 emorag emorag    7 abr 10 19:41  fichero_7.sh
-rwSrw-r-- 1 emorag emorag  216 abr 14 13:42  mio.c
-rw-rw-r-- 1 hacker emorag  216 abr 14 13:34  otroyo
drwx------ 2 emorag emorag 4096 abr 14 13:38  solomio
total 20
-rwxrwxr-x 1 emorag emorag 8440 abr 14 13:38 a.out
-rw-rw-r-- 1 emorag emorag    0 abr 14 13:20 fich1
-rw-rw-r-- 1 emorag emorag    0 abr 14 13:20 fich2
-rw-rw-r-- 1 emorag emorag  216 abr 14 13:42 mio.c
-rwSrw-r-- 1 emorag emorag  216 abr 14 13:37 otroyo.c
```

En este caso el usuario efectivo y real del archivo otroyo es distinto, pero mio es el mismo.

<u>Ejercicio 15. Copia en tu directorio el fichero el fichero /bin/sh (del intérprete de ordenes) como hsh y como propiedad de hacker y SUID. Ejecútalo y comprueba que ocurre con las ordenes, whoami, ls, ls ../hacker y exit. Interpreta el resultado.</u>

```
emorag@emorag:~/Escritorio/PRACTICA$ sudo cp /bin/sh hsh
emorag@emorag:~/Escritorio/PRACTICA$ sudo chown hacker hsh
emorag@emorag:~/Escritorio/PRACTICA$ sudo chmod u+s hsh
emorag@emorag:~/Escritorio/PRACTICA$
emorag@emorag:~/Escritorio/PRACTICA$
emorag@emorag:~/Escritorio/PRACTICA$ ./hsh
$ whoami
hacker
$ ls
 a.out            Ejercicio_4.sh    EJERCICIO_8    fichero_3.sh   fichero_6.sh   mio.c
 Ejercicio_13.sh  Ejercicio_5.sh    fichero_1.sh   fichero_4.sh   fichero_7.sh   otroyo
'EJERCICIO 3'     EJERCICIO_7.txt   fichero_2.sh   fichero_5.sh   hsh            solomio
$ ls ../hacker
ls: no se puede acceder a '../hacker': No existe el archivo o el directorio
$ exit
emorag@emorag:~/Escritorio/PRACTICA$
```

El primer comando dice quien es el usuario, el segundo muestra todo lo que ve el usuario, el tercero no funciona y el ultimo se sale.

Ejercicio 16. Estudiar el manual de la orden write. Estudiar lo que ocurre cuando las líneas escritas comienzan con el carácter ¿Supone esto algún peligro para el sistema? Estudiar los permisos del programa /bin/write.

Si ! (la exclamación cerrada) se encuentra en el inicio de la línea de comando, el comando write llama a la Shell para ejecutar el resto de la línea de comando.

```
emorag@emorag:~/Escritorio/PRACTICA$ ls -l /usr/bin/write
lrwxrwxrwx 1 root root 23 nov  4 08:33 /usr/bin/write -> /etc/alternatives/write
```

Es peligroso porque al tener permisos de root puede ejecutar cualquier comando que se pase por la línea de comando.

Ejercicio 17. Revisar las ayudas de las dos órdenes getfacl y setfacl e identificar las diferentes opciones en los ejemplos de modificación de permisos mediante setfacl:

setfacl –b –k –R *

setfacl –R –m g:sistemas:rw

Con el getfacl lo que hacemos es obtener listas de control de acceso a archivos y con el setfacl lo que hacemos es establecer listas de control de acceso a archivos.

El primer comando tiene las opciones:

- ✓ -b: elimina todas las entradas ACL extendidas.
- ✓ -k: elimina el ACL predeterminado, establece entradas ACL a establecer desde “arch”.
- ✓ -R: recorre los directorios de manera recursiva.

Ese primer comando ejecuta todas esas opciones sobre todos los directorios de manera recursiva.

El segundo comando tiene las opciones:

- ✓ -R: lee entradas ACL desde “arch”.
- ✓ -m: modifica ACL actual de archivo.

Ese segundo comando ejecuta estas dos opciones sobre el fichero al que llama, es decir modifica el ACL del archivo actual por el de g:sistemas:rw.

Ejercicio 18. Escribir los ACLs del resto de ficheros y directorios listados en el ejemplo anterior y verificar los valores efectivos debidos a mask.

```
emorag@emorag:~/Escritorio/PRACTICA$ sudo setfacl -m u::rwx,u:emorag1:rwx,emorag2:r-x,g::r-x,g:e
morag2:r--,m::-w-,o::r-x ../PRACTICA
```

Ejercicio 19. Escribir la orden para conceder permisos de escritura a un usuario de tu grupo sobre el fichero datos.

```
emorag@emorag:~/Escritorio/PRACTICA$ setfacl -m u:emorag:r fichero.txt
emorag@emorag:~/Escritorio/PRACTICA$ getfacl fichero.txt
# file: fichero.txt
# owner: emorag
# group: emorag
user::rwx
user:emorag:r--
group::rwx
mask::rwx
other::r--
```

Ejercicio 20. ¿Si el usuario pedro perteneciese al grupo bin, podría ejecutar el fichero del ejemplo anterior? Si la ACL del fichero fuese la siguiente:

```
1  # file: datos
2  # owner: pedro
3  # group: inf
4  user::rw-
5  user:pedro:rw-
6  group::r-x
7  other::---
```

A) ¿Podría el usuario pedro en el grupo inf ejecutar el fichero?
No, porque los permisos de pedro en el grupo inf son: rw- según la sentencia de la línea 5.

B) ¿Podría el usuario pedro en el grupo bin ejecutar el fichero?
No, porque según la sentencia de la línea 4, que sería un usuario específico en un grupo cualquiera.

C) ¿Qué permisos tendría el usuario pedro en el grupo inf?
Según la sentencia de la línea 5, pedro en el grupo inf tiene los permisos de rw-

D) ¿Qué permisos tendría el usuario juan en en grupo bin?
Según la sentencia de la línea 7, juan no tiene ningún permiso en el grupo bin.

Ejercicio 21. Probar la orden chacl en distintos directorios del sistema. Utiliza la opción -l para comparar los dos formatos de salida.

```
emorag@emorag:~/Escritorio/PRACTICA$ chacl u::rwx,g::r-x,o::r-- fichero.txt
emorag@emorag:~/Escritorio/PRACTICA$ chacl -l fichero.txt
fichero.txt [u::rwx,g::r-x,o::r--]
```

```
emorag@emorag:~/Escritorio/PRACTICA$ chacl -l /bin/
/bin/ [u::rwx,g::r-x,o::r-x]
emorag@emorag:~/Escritorio/PRACTICA$ chacl -l /tmp/
/tmp/ [u::rwx,g::rwx,o::rwx]
```

Ejercicio 22. Crea un fichero cualquiera y concede los permisos que quieras a un usuario de tu grupo y a uno de un grupo distinto.

```
emorag@emorag:~/Escritorio/PRACTICA$ getfacl fichero.txt
# file: fichero.txt
# owner: emorag
# group: emorag
user::rwx
group::r-x
other::r--

emorag@emorag:~/Escritorio/PRACTICA$ setfacl -m u:emorag:x fichero.txt
emorag@emorag:~/Escritorio/PRACTICA$ getfacl fichero.txt
# file: fichero.txt
# owner: emorag
# group: emorag
user::rwx
user:emorag:--x
group::r-x
mask::r-x
other::r--
```

A) Salvaguarda la ACL del fichero copiándola sobre otro.

```
emorag@emorag:~/Escritorio/PRACTICA$ getfacl fichero.txt | setfacl --set-file=- fichero2.txt
emorag@emorag:~/Escritorio/PRACTICA$ getfacl fichero2.txt
# file: fichero2.txt
# owner: emorag
# group: emorag
user::rwx
user:emorag:--x
group::r-x
mask::r-x
other::r--
```

B) Cambia los permisos del fichero usando la orden chmod. ¿Qué ocurre con la ACL asociada?

```
emorag@emorag:~/Escritorio/PRACTICA$ chmod 222 fichero.txt
emorag@emorag:~/Escritorio/PRACTICA$ getfacl fichero.txt
# file: fichero.txt
# owner: emorag
# group: emorag
user::-w-
user:emorag:--x                 #effective:---
group::r-x                      #effective:---
mask::-w-
other::-w-
```

He intentado cambiar los permisos a 222, los de user y other han cambiado bien a rwx, pero los de group no, en cambio los de mask se han puesto a rwx.