Apuntes y órdenes (Prácticas: Módulo I)

Sesión 1

- Órdenes para gestión de cuentas de Usuario

whoami → Devuelve el nombre del usuario actual
useradd <NombreUsuario>
ó → Añade un usuario al /home
adduser <NombreUsuario>
usermod → Modifica una cuenta ya existente
userdel → Elimina una cuenta pero no borra el directorio home
newusers → Usa un .txt con el formato de /etc/passwd para crear cuentas de usuario
passwd → Cambia la contraseña de un usuario
chsh -s <shell> <usuario> → Cambia el intérprete de órdenes por defecto (/etc/passwd) con esta opción también podemos establecer como shell un archivo ejecutable.

- Contenido de archivos de /etc

/etc/shadow

Mike:!!:17824:0:99999:7:::
migue:Contr:17824:0:99999:7:::

<nombre> : <password cifrado> : <1> : <2> : <3> : <4> : <5> : <6>

1. Días transcurridos desde 1-1-1970 donde el password fue cambiado por última vez (por defecto).
2. El mínimo número de días entre cambios de contraseña.
3. Días máximos de validez de la cuenta.
4. Días que avisa antes de caducar la contraseña.
5. Días después de que un password caduque para deshabilitar la cuenta
6. Fecha de caducidad. días desde 1-1-1970, donde la cuenta es deshabilitada y el usuario no podrá iniciar sesión

/etc/passwd

root::0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
Mike:x:500:500::/home/Mike:/bin/bash
migue:x:501:501::/home/migue:/bin/bash

<nombre> : <password> : <uid> : <gid> : <descripción opcional> : <carpeta> : <shell>

uid → Número identificador del usuario (entre 0 [root] y 65535)
gid → Grupo identificador (cada user tiene un grupo principal pero puede pertenecer a más)
carpeta → Donde se encuentra el usuario
shell → Es el intérprete de órdenes que se ejecuta por defecto cada vez que entra al sistema. Los users con permisos limitados no deben tener shell, se les deja con /usr/bin/nologin o /bin/false.

/etc/group

root:x:0:root
bin:x:1:root,bin,daemon
Mike:x:500:
migue:x:501:

<nombre grupo> : <password grupo> : <gid> : <miembros del grupo>

password → Si aparece una x es que está encriptada.

- **Acceso a información del SO relativa a sistemas de archivos**

Los archivos /etc/fstab y /etc/mtab muestran información sobre los sistemas de archivos que se encuentran montados en el sistema. /etc/fstab es muy útil para comprender las opciones de montaje

Sesión 2

- Partición de dispositivos de almacenamiento secundario

Al proceso de establecer e identificar estas particiones en el dispositivo se le denomina comúnmente partición de disco. Cuando creamos una partición es necesario asociarle una etiqueta que indica el tipo de SA que va a alojar cuando posteriormente se formatee. Esta información se almacena mediante un código numérico que determina el tipo de partición. Con **/sbin/sfdisk -T** obtenemos una lista de dichos códigos. Linux limita el número máximo de particiones que se pueden realizar sobre un disco SCSI a 15, y a un total de 63 sobre un disco IDE.

mknod → Te permite hacer archivos especiales de bloques y de caracteres. Nos servirá para preparar un dispositivo simulado. Explicación de uso en sesión 3.

dd → Copia RAW DATA (bajo nivel) y lo convierte según los operandos que le añadamos
dd if=<Input File> of=<Output File> bs=<Nº Bytes> count=<Nº Copias>
Por ejemplo si queremos un archivo vacío de 20MB podemos utilizar como Input File el /dev/zero que es un archivo especial que provee tantos caracteres *null* como se lea de él, entonces si ponemos **dd if=/dev/zero of=<archivo> bs=2k count=10000** obtendremos un archivo de 20MB con el nombre <archivo> vacío. También podemos usar otro bs / count, eso se especifica en el man. (Con esa orden lo que hemos hecho ha sido coger ceros a nivel de bloques de 2k y 10000 veces, es decir 20MB)

losetup → Asocia un archivo de dispositivo loop a otro archivo, así el "disco virtual" del archivo pasa a estar asociado al loop, es decir toda la memoria que hayamos creado previamente para el archivo se asocia al dispositivo loop.
losetup <dispositivo loop> <archivo>
losetup -d <dispositivo loop> → Elimina el dispositivo

fdisk <dispositivo> → Manipula la tabla de particiones, abre un menú con opciones, con m podemos ver las distintas opciones, de momento hemos usado n (añade una nueva partición), w (escribe la tabla y se sale, si no hacemos esto se pierden los datos pues antes de usar w están en la RAM).

fdisk >> n >> e \ p e extended \ p primary(1-4), como solo queremos una entrada hacemos primary para usar todo ese espacio como un "pen-drive" virtual

fdisk -l <dispositivo> → Te da info de la partición, nos sale esto:

Device Boot	Start	End	Blocks	Id	System
/dev/loop0p1	2048	39999	18976	83	Linux

- Formateo lógico de particiones

mke2fs → Create an ext2/ext3/ext4 filesystem

Para reproducir la salida que aparece en el guión de prácticas he usado la orden
mke2fs -j /dev/loop0 -L LABEL_ext3
ó mkfs.ext3 -L LABEL_ext3 /dev/loop0

tune2fs → Sirve para modificar algunos parámetros de los sistemas de archivos.

tune2fs <parámetro u opciones> <SA>

Con **tune2fs -l <SA>** obtenemos información

- ¿Cómo podrías conseguir reservar para uso exclusivo de un usuario username un número de bloques del sistema de archivos?

tune2fs [sistema de archivos] –r [número] –u [username]

- Montaje y desmontaje de Sistemas de Archivos

mount → Te permite ampliar el espacio de nombres añadiendo una nueva rama, de manera

que toda la información del sistema de archivos montado será accesible en el espacio de nombres. (Podríamos crear nuevos archivos y directorios en esta rama). Antes de montar un dispositivo en una dirección esta debe existir, podemos crearla con mkdir.

mount <dispositivo> <dirección>

umount → Desmonta los sistemas de archivos, si usamos -d servirá para los loops

Para desmontar usamos umount y el loop /dev/loop0 por ejemplo, así para desmontar el disco que hemos montado pondríamos:

umount /dev/loop0 Si pusiéramos umount /mnt/lost+found no funcionaría

umount <dispositivo loop>

Archivo /etc/fstab

Es el archivo de configuración que contiene la información sobre todos los sistemas de archivos que se pueden montar y de las zonas de intercambio a activar. El formato es:

<file system>	**<mount point>**	**<type>**	**<options>**	**<dump>**	**<pass>**
LABEL=ROOT	/	auto	noatime	1	1
tmpfs	/dev/shm	tmpfs	defaults	0	0
tmp	/tmp	tmpfs	rw,mode=1777...	0	0
devpts	/dev/pts	devpts	gid=5,mode=620	0	0
sysfs	/sys	sysfs	defaults	0	0
proc	/proc	proc	defaults	0	0

Podemos añadir líneas para que los SA se monten automáticamente en el arranque:

<file system>	**<mount point>**	**<type>**	**<options>**	**<dump>**	**<pass>**
/dev/loop0	/mnt/SA_ext3	ext3	auto,ro	0	0
/dev/loop1	/mnt/LABEL_ext4	ext4	auto,dirsync	0	0

Esto haría que el /dev/loop0 fuese de tipo ext3, se montara en el directorio /mnt/SA_ext3 y fuese sólo de lectura. El /dev/loop1 fuese de tipo ext4, se montara en /mnt/LABEL_ext4 y tuviese sincronizadas sus operaciones de E/S de modificación de directorios.

Sesión 3

- Órdenes para control y gestión de CPU

uptime → Muestra una línea con la siguiente información: la hora actual, el tiempo que lleva en marcha el sistema, el número de usuarios conectados, y la carga media del sistema en los últimos 1, 5 y 15 minutos.

! Las options del uptime no se pueden usar en el root, sólo en la terminal normal.

```
17:51:13 up  7:20,  1 user,  load average: 0,35, 0,57, 0,80
```

w → Igual que uptime pero desarrolla los usuarios conectados.

```
17:51:15 up  7:20,  1 user,  load average: 0,32, 0,56, 0,79
USUARIO  TTY      DE       LOGIN@   IDLE      JCPU      PCPU WHAT
mike     :0       :0       12:31    ?xdm?     5:04      0.01s /usr/lib/gdm3/gdm...
```

time → Mide el tiempo de ejecución de un programa y muestra un resumen del uso de los recursos del sistema.

Tiempo de Espera = real – user - sys

time <orden> → Te muestra la ejecución de la orden y luego el tiempo que tarda medido con time.

ps → Te muestra información de los procesos activos

-auroot → Muestra los procesos del root

-auuser → Muestra los procesos del usuario "user"

-ef → Muestra información completa de todos los procesos del sistema

$: time ps

```
  PID    TTY          TIME CMD
 7559    pts/2    00:00:00 bash          → orden (ps)
13958    pts/2    00:00:00 ps

real    0m0,019s
user    0m0,004s                        → time
sys     0m0,015s
```

Info

PID → Process ID

TTY → Terminal asociada al proceso

TIME → Tiempo en el formato hh::mm::ss

CMD → Nombre del ejecutable

nice → Ejecuta un programa modificando su prioridad. Rango de valores [-20, 19]

Ej: **nice -número programa** → Disminuye su valor de prioridad

Ej: **nice --número programa** → Aumenta su valor de prioridad

! El usuario SOLO PUEDE DISMINUIR la prioridad, el root puede aumentar y disminuir.

! Valor de prioridades más pequeño equivale a una prioridad mayor

renice → Modificamos la prioridad del proceso usando su número de proceso

Ej: **renice 15 4332** → Pone el valor de prioridad a 15 del proceso 4332

En PID vemos el número de
proceso, en la columna de PR vemos el valor de la prioridad y en NI el cambio en PR

! Ejecutar en background --> ./programa &

pstree → Visualiza el árbol de procesos en ejecución.

top → Proporciona una visión continuada de la actividad del procesador en tiempo real
- En PID vemos el número de proceso, en la columna de PR vemos el valor de la prioridad y en NI el cambio en PR

mpstat → Muestra las estadísticas del procesador del sistema junto con la media global de todos los datos mostrados
mpstat <intervalo de tiempo> <número de muestreos>

Cuestiones:

- Saber cuánto tiempo lleva en marcha el sistema
  - Con uptime en la parte que pone "up to x min".
- Cuántos usuarios hay conectados
  - Tanto uptime como w, pero w te especifica quienes son.
- Cuál es la carga media del sistema en los últimos 15 minutos
  - uptime << load average: 1min 5min 15min

- Crea un script o guión shell que realice un ciclo de un número variable de iteraciones en el que se hagan dos cosas: una operación aritmética y el incremento de una variable. Cuando terminen las iteraciones escribirá en pantalla un mensaje indicando el valor actual de la variable. Este guión debe tener un argumento que es el número de iteraciones que va a realizar. Por ejemplo, si el script se llama prueba_procesos, ejecutaríamos: prueba_procesos 1000

```
#!/bin/bash
x=0
y=1
for i in `seq 1 $1`
do
y=$(($y*2))
x=`expr $x + 1`
done
echo “El valor de la variable es “ $x
```

- Ejecuta el guión anterior varias veces en background (segundo plano) y comprueba su prioridad inicial. Cambia la prioridad de dos de ellos, a uno se la aumentas y a otro se la disminuyes, ¿cómo se comporta el sistema para estos procesos?

```
$: ./prueba_procesos 30000 &
[1] 15594
$: ./prueba_procesos 40000 &
[2] 15896
```

La prioridad inicial es de 20 para ambos (por defecto)

#: renice -5 15594 → Resta 5 a la prioridad del proceso

#: renice 25 15896 → Suma 19 (MÁX) a la prioridad del proceso

- Ejecuta la orden ps con la opción -A, ¿qué significa que un proceso tenga un carácter "?" en la columna etiquetada como TTY?

  Que no tiene ningún terminal en concreto asociado

- Órdenes para control y gestión de memoria

free → Parecido a top pero mucho más ligera (consume menos recursos - CPU y memoria)

watch → Ejecuta una orden cada 2s (por defecto), con -d puedes ver las diferencias resaltadas.

vmstat → Supervisar el sistema mostrando información de memoria, procesos, E/S y CPU.

- Órdenes para control y gestión de E/S

du <directorio> → Contabiliza el número de bloques de datos está usando. Para que te de un formato que podamos entender (human readable format) tenemos que poner -d

df → Contabiliza el número de bloques disponibles en disco, si no se da ningún argumento aparece por defecto el espacio disponible en todos los sistemas de archivos montados. -i lista de inodos | -h en modo entendible

ln <archivo base> <archivo salida> → Linka dos archivos e incrementa el contador de enlaces (ls -lai 3ª columna, antes del user). Por defecto es un hardlink, pero si añadimos la opción -s, se convierte en un softlink (no contabiliza en el contador de enlaces)

mknod → Crea archivos especiales de bloques o caracteres, esta orden permite especificar el major (determina el controlador al que está conectado el dispositivo) y el minor (el dispositivo en sí)

mknod <nombre archivo> b <major> <minor> → Archivo de bloque

mknod <nombre archivo> c <major> <minor> → Archivo de caracteres

Con la orden ls -l podemos ver el tipo de archivo **b**rw-r--r-- **c**rw-r--r--

! El minor debe ser el mismo número que el del dispositivo, es decir si hacemos un mknod /dev/loop**1** el minor debe ser un **1**

Sesión 4

- Los procesos demonio

Un proceso demonio es un proceso con vida propia, independiente y que se ejecuta en un plano invisible. En muchos casos se ejecutan con privilegio de superusuario (UID=0) y tienen por padre al proceso init (PPID=1).

- Órdenes con periodicidad (at, batch, crontab)

at → Lee órdenes de la entrada estándar o de un script y lo ejecuta a una hora concreta (SOLO UNA VEZ)

at -f fichero <hora> En fichero metemos el script

Algunas órdenes de at:

at -f fichero midnight	→ A medianoche de hoy
at -f fichero midnight+1 minute	→ Un minuto después de medianoche
at -f fichero 17:30 tomorrow	→ A las 17:30 de mañana
at -f fichero Dec 25 2019	→ Hora actual pero del 25 Diciembre
at -f fichero midnight Jan 01 2019	→ A las 00:00 del 1 de enero del 2019

atq → Muestra la cola de procesos que se ejecutarán y a la hora.

<Nº Proceso (ID)> <Día de Semana> <Mes> <Día de Mes> <Hora> <Año> <User>

3 Tue Jan 1 00:00:00 2019 a mike

! Si lo ejecuta un usuario sólo podrá ver su cola de procesos, si lo hace un root verá todos.

atrm <Nº Proceso> → Elimina el proceso del que pasamos su ID

Ejemplos:

```
#!/bin/bash
#Nombre: script_4.7.sh

at -q c -f ./script_4.4.sh now+1 minute
at -q d -f ./script_4.4.sh now+1 minute
at -q e -f ./script_4.4.sh now+1 minute
---------------------------------
$ atq -q c
$ atq -q d
$ atq -q e
```

batch → Parecido a at pero sin especificar la hora

crontab → Ejecuta órdenes con una periodicidad, usa el demonio cron

```
# minute  hour  day_of_month  month  day_of_week  <orden o script>
```

minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names, see below)
day of week	0-7 (0 or 7 is Sun, or use names)

Ejemplo de uso de crontab

Contenido de crontab_file:

*/1 * * * * ~/Escritorio/script.sh #← Ejecuta el script cada minuto

Contenido del script:

```
#!/bin/sh                    #← Hay que poner esto
ls -l ~ > ~/Escritorio/archivo_prueba.txt
echo "Hecho `date +%Y-%j-%T`" >> ~/Escritorio/archivo_prueba.txt
```

Lo ejecutaríamos con: “crontab crontab_file”

- Otras cosas que habría que saber

Cargar home:

mount none / -t hostfs -o /fenix/alum/d1/miguealguacil
--> Te lo copia en /root

grep → Sirve para buscar una cadena.

ps -ef | grep daemon --> Busca "daemon" en la salida de ps -ef

usos del man

man comando man 4 comando (va a la página 4 del comando)

Nombre de archivos {absoluto, relativo}

cd change directory → cwd (current working directory) ~ esta es la forma relativa

Restricción para evitar ambigüedad → No puedes poner el mismo nombre de archivo

filename → Trozo que se aloja en la entrada a directorio, sirve para identificar un archivo en un directorio

pathname → En el espacio de nombres

inodos → Estructuras de metadatos que soporta el archivo(?)

con ls -li podemos verlos, sería el **14239**

```
-rw-r--r-- 1 root root 20480000 Oct  1 06:33 archivo_SA20      <-- ls -l
14239 -rw-r--r-- 1 root root 20480000 Oct  1 06:33 archivo_SA20      <-- ls -li
```