

## Índice

Sesión 1. Herramientas de administración básicas.....	1
Actividades Sesión 1:.....	9
Repaso Sesión 1:.....	12
Sesión 2. Herramientas de administración del SA.....	12
Actividades Sesión 2:.....	18
Repaso Sesión 2:.....	18
Sesión 3. Monitorización del sistema.....	19
Actividades Sesión 3:.....	28
Repaso Sesión 3:.....	32
Sesión 4. Automatización de tareas.....	33
Actividades Sesión 4:.....	40
Examen 2019-2020.....	44

## Sesión 1. Herramientas de administración básicas

PARA MOSTRAR INFORMACIÓN EN LA TERMINAL DE CUALQUIER COMANDO USAR:  
**[Comando] –help** → Mostrará información en ingles así como las diversas opciones y modos de uso.

PARA LEER UN FICHERO: **cat [DIRECTORY]** → Mostrará la información que guarde dicho directorio. Uno con el que se trabajará bastante será */etc/passwd*.

**Ejemplo:** `cat /etc/passwd`

### USUARIO ROOT

También es llamado superusuario.

Un usuario root es aquel que cuenta con derechos de administrador. Tiene acceso a toda la memoria.

**\*Nota:** Cuando estemos trabajando con un sistema Linux/UNIX, por seguridad, deberíamos hacerlo siempre bajo una cuenta de usuario normal y no usar la cuenta del administrador, del root, a no ser que queramos realizar tareas de administración. Debemos tener en cuenta que el administrador es el usuario con el mayor privilegio.

El usuario identificado con nombre de *usuario root* pertenece al *grupo root* y su *directorio* home es */root*.

**whoami** -> Te indica el usuario actual

Si ya se ha iniciado un shell con otro usuario:

**su** -> Pide un cambio de usuario a root

**su [USER]** -> cambia al usuario mencionado

**UID** es el número entero que identifica (id) al usuario. El SO sabe quien es el user por el UID.

**GID** es el grupo al que pertenece el usuario. El de root es el 0

**id** -> este comando muestra el UID, GID y grupos del usuario actual

**usermod** -> cambia el UID de u usuario. Ejemplo: **usermod -u 1004 user\_2** (debemos ser superusuario con permisos de ejecución)

etc/passwd -> Almacena información de las cuentas de usuarios

etc/shadow -> Guarda los password encriptados e información de "envejecimiento" de las cuentas

etc/group -> Definición de los grupos y usuarios miembros

## **GESTIÓN DE USUARIOS**

### **a) CREACIÓN DE USUARIO**

**useradd [nombreUsuario]**

Para comprobar que se creo, navegar hasta la carpeta home:

**\*Nota para navegar:**

**cd ..** (vuelve atrás)

**cd [nombreCarpeta]** (navega a carpeta nombreCarpeta) -> **cd home**

**ls** (lista aquello que haya en el directorio)

**\*Nota:** Para listar los usuarios introducir el comando **cat /etc/passwd** y se mostrará una lista.

**useradd -p [PASSWORD] [USER]** → crea un usuario con una contraseña directamente.

### **MODIFICAR VALORES DEL USUARIO**

- **usermod** → modifica una cuenta de usuario ya existente

**usermod [options] [user]** → **usermod -p nuevaPass Prueba1** → cambia la contraseña.

- **userdel** → elimina una cuenta de usuario (pero no borra el directorio en home, por defecto).

- **newusers** -> crea cuentas de usuario utilizando la información de un archivo de texto con el mismo formato que /etc/passwd.

- **system-config-users** -> Herramienta de gestión de usuarios en modo grafico (No sirve en el Kernel de Fedora).

**\*Nota:** Si se pone en la terminal el comando sin nada, sale un desplegable de opciones para usarlo. Por ejemplo, en el caso de **usermod -p, --password PASSWORD** indica algo más después en mayúsculas, quiere decir que después de usar la opción hay que poner un valor antes del USER.

En el directorio `/etc/skel` se guardan unos archivos de configuración de shell, los cual se copian al directorio HOME asignado cuando se crea una cuenta de usuario. Estos archivos en bash son:

- **`.bash_profile`** → se ejecuta al hacer el login (conectarse al sistema) y en él podremos indicar alias, variables, configuración del entorno, etc. que deseamos iniciar al principio de la sesión.

- **`bashrc`** → su contenido se ejecuta cada vez que se ejecuta una shell, tradicionalmente en este archivo se indican los programas o scripts a ejecutar.

- **`.bash_logout`** → se ejecuta al salir el usuario del sistema y en él podremos indicar acciones, programas, scripts, etc., que deseamos ejecutar al salirnos de la sesión.

## **b) CAMBIAR CONTRASEÑA (PASSWORD)**

**`passwd [USER]`**

**\*Nota:** Al usar este comando si no se indica USER se entiende que se quiere cambiar la del usuario actual. El root puede cambiar cualquier contraseña, mientras que un usuario normal solo la suya propia.

**`chsh`** → podemos cambiar el intérprete de órdenes por defecto que usará el usuario cuando se conecte al sistema. En el último campo de `/etc/passwd` se establece para cada usuario el intérprete de órdenes que se ejecuta por defecto cada vez que entra al sistema. En el archivo `/etc/shells` se indican los shells permitidos.

## **c) Parámetros de configuración de una cuenta**

Para las cuentas de los usuarios se pueden establecer restricciones de tiempo, también llamadas de envejecimiento, respecto a la validez de la cuenta o la contraseña. Los valores se guardan en el archivo `/etc/shadow`.

- **`changed`** → fecha del último cambio de contraseña.

- **`minlife`** → número de días que ha de pasar para poder cambiar la contraseña.

- **`maxlife`** → número de días máximo que puede estar con la misma contraseña sin cambiarla.

- **`warn`** → cuántos días antes de que la contraseña expire (`maxlife`) será informado sobre ello.

- **`inactive`** → número de días después de que la contraseña expire que la cuenta se deshabilitará de forma automática si no ha sido cambiada.

- **`expired`** → fecha en la que la cuenta expira y se deshabilita de forma automática.

- **chage** → permite la modificación de los parámetros de la cuenta. Otra alternativa es **passwd**. LOS VALORES POR DEFECTO ESTÁN EN: */etc/login.defs*

**chage [options] [USER]**

Opciones:

- **chage -d ult\_día usuario** → fecha del último cambio de password.
- **chage -m min\_días usuario** → número de días que han de pasar para poder cambiar la contraseña.
- **chage -M max\_días usuario** → número de días máximo que puede estar con la misma contraseña sin cambiarla.
- **chage -W warn\_días usuario** → modificar el valor de warn.
- **chage -I inac\_días usuario** → modificar el valor de inactive.
- **chage -E exp\_días usuario** → modifica el valor de expired.
- **chage -l -list** → muestra información de antigüedad de la cuenta.

#### **d) Gestión de grupos**

Un grupo es un conjunto de usuarios que comparten recursos o archivos del sistema. Con los grupos se pueden garantizar permisos concretos para un conjunto de usuarios, sin tener que repetirlos cada vez que se desee aplicarlos.

Los grupos creados se guardan en el fichero */etc/group*

**root:x:0:root**

**bin:x:1:root,bin,daemon**

**daemon:x:2:root,bin,daemon...**

Consta de 4 campos separados por : (dos puntos) que indican:

- 1.- Nombre del Grupo
- 2.- La contraseña del grupo (cuando aparece una x indica que existe un archivo gshadow)
- 3.- El GID (número identificador único del grupo)
- 4.- Lista de los miembros del grupo separados por comas y sin espacios

Opciones relacionadas con la gestión de los grupos:

- **groupadd [options] [NOMBREGRUPO]** → crea un nuevo grupo.
- **groupmod [options] [NOMBREGRUPO]** → modifica un grupo existente.
- **groupdel [NOMBREGRUPO]** → elimina un grupo.
- **newgrp [NOMBREGRUPO]** → cambia de grupo activo (lanza un shell con ese grupo),
- **gpasswd [NOMBREGRUPO]** → asigna una contraseña a un grupo.
- **gpasswd -a [USER] [NOMBREGRUPO]** → añade un usuario a un grupo.
- **groups [USER]** → informa de los grupos a los que pertenece un usuario.
- **id [USER]** → lista el identificador del usuario y los grupos a los que pertenece.
- **grpck -->** comprueba la consistencia del archivo de grupos.

## USUARIOS Y GRUPOS ESPECIALES

Los usuarios especiales son aquellos que no están asociados a personas físicas.

Usuarios especiales del sistema:

- **root** → Usuario administrador del sistema.
- **bin, daemon, lp, sync, shutdown,...** → Tradicionalmente usados para poseer archivos o ejecutar servicios.
- **mail, news, ftp,...** → Asociados con herramientas o utilidades.
- **postgres, mysql, xfs,...** → Creados por herramientas instaladas en el sistema para administrar y ejecutar sus servicios.
- **nobody ó nfsnobody** → Usada por NFS y otras utilidades.

Grupos estándar del sistema:

- **root, sys, bin, daemon, adm, lp, disk, mail, ftp, nobody** → Algunos de los nombres de grupo preconfigurados por los sistemas UNIX. Los GIDs inferiores a 500 están reservados para estos grupos.
- **tty, dialout, disk, cdrom, audio, video** → Nombres de dispositivos.
- **Kernel** → Grupo propietario de los programas para leer la memoria del kernel.
- **Users** → Puede usarse como grupo por defecto para todos los usuarios normales del sistema.

## Organización del sistema de archivos y gestión básica de archivos

La organización de los archivos en un sistema de archivos presenta una estructura jerárquica en forma de árbol en donde los nodos interiores (internos) están representados por los directorios y los nodos finales (nodos hoja) están representados por los archivos asociados con un directorio. Podemos tener un directorio que no contenga ningún archivo, por lo que sería un nodo hoja (pero no tendría sentido).

Un archivo, en la estructura jerárquica de directorios/archivos puede ser referenciado (nombrado) de dos formas distintas: de manera absoluta (su nombre empieza por "/" en sistemas UNIX), o de manera relativa (su nombre no empieza por "/").

Comando **find** → busca archivos en el sistema de archivos.

- **find /DIR/DIR -name “\*.EXTENSIÓN”** → para buscar todos los archivos dentro de un directorio con una extensión.

- **find / -name [NOMBREARCHIVO]** → para encontrar un archivo en todo el sistema, podemos buscarlo desde la carpeta raíz /

- **find / -user [NOMBREUSUARIO]** → si deseamos buscar un archivo por nombre de usuario. Esto mostrará todos los archivos de ese usuario a partir de la carpeta raíz / y todas sus subcarpetas.

- **find . -size 0c** → para buscar archivos vacíos (el . Indica el directorio en el que nos encontramos).

- **find . -size 54k** → para buscar archivos de un tamaño en kilobytes

- **find . -regex ‘./archivo0[1-2]\_0[1-3].\*’** → también podemos buscar utilizando expresiones regulares. Este ejemplo en concreto buscará archivos del tipo archivo01\_01.txt, archivo02\_03.txt...

Comando **locate** → busca archivos en el índice del sistema, devolviendo todas las rutas donde aparece el nombre buscado:

**locate [NOMBREARCHIVO]**

Y podemos consultar dentro de los resultados devueltos, por ejemplo:

**locate [ELARCHIVO] | grep bin** → devolverá todas las rutas del sistema que contengan el nombre [ELARCHIVO] y que además aparezca el texto bin.

**\*Nota:** es recomendable actualizar el índice del sistema con `sudo updatedb` para añadir las rutas recientes.

### **Organización común en sistemas de archivos tipo Linux. Filesystem Hierarchy Standard (FHS)**

El FHS es un estándar que propone una forma sistemática de organizar toda la información que almacena un sistema operativo tipo Linux. Uno de los directorios importantes recogidos en el estándar FHS es el directorio /etc. Básicamente en este directorio podemos encontrar todos los archivos de configuración del sistema.

Directorios que recoge FHS:

**/bin** → Programas de utilidad fundamentales para ser utilizados por cualquier usuario del sistema.

**/sbin** → Programas de utilidad fundamentales para ser utilizados por el usuario root.

**/boot** → Archivos fundamentales para el programa Boot Loader.

**/dev** → Todos los archivos especiales de dispositivo.

**/etc** → Archivos de configuración del sistema.

**/home** → Los directorios de inicio de todos los usuarios que disfrutan de una cuenta en el sistema, excepto, el directorio de inicio del root: /root

**/lib** → Bibliotecas sin las que no pueden funcionar los programas ubicados en /bin y /sbin.

**/media** → Este directorio actúa como punto de montaje para dispositivos extraíbles: DVD-ROM, dispositivos USB, etc.

**/mnt** → Este directorio actúa como punto de montaje para sistemas de archivos montados temporalmente.

**/opt** → Normalmente aquí se ubican los programas que no forman parte de la distribución instalada en el sistema.

**/proc** → Sistema de archivos virtual que hace de interfaz con el núcleo y los procesos.

**/tmp** → Archivos temporales que normalmente no se mantienen una vez se apaga el sistema.

**/usr** → Archivos ejecutables, archivos de código fuente, bibliotecas, documentación y, en general, todos los programas y utilidades.

**/var** → Los archivos cuyo contenido se espera que cambie durante el funcionamiento normal del sistema.

## **Órdenes básicas para gestión del sistema de archivos**

### **Comandos para explorar la estructura jerárquica del directorio/archivos:**

- **pwd** → muestra el nombre del directorio de trabajo actual.
  - **pwd -L** → valor de la variable \$PWD.
  - **pwd -P** → muestra el directorio físico sin enlaces simbólicos.
- **ls** → muestra información de los archivos y directorios en el directorio actual.
  - **ls -l** → muestra toda la información de los ficheros (atributos, usuarios, etc.).
  - **ls -a** → muestra también los archivos ocultos.

### **Comandos para crear y borrar directorios y archivos:**

- **mkdir [options] [DIRECTORIO]** → Crea un nuevo directorio.
- **rmdir [options] [DIRECTORIO]** → Borra un directorio (si no está vacío).
- **cat [options] [NOMBREFICHERO/RUTAFICHERO]** → Muestra la estructura de un archivo.
- **rm [options] [NOMBREFICHERO]** → Borra un archivo (o desenlaza los ficheros).

### **Comandos para copiar y mover archivos y directorios a distintas ubicaciones:**

- **cp** → Copia el contenido fuente en un directorio destino.

- **cp [options] [-T] [FUENTE] [DESTINO]**
- **cp [options] [FUENTE1] [FUENTE2]... [DESTINO]**
- **cp [options] -t [DESTINO] [FUENTE1] [FUENTE2]...**

*cp main.c def.h /home/josu/practicas -> copia ambos archivos en practicas*  
*cp \*.c backup -> copia todos los archivos.c en el directorio backup*

- **mv [options] [-T] [FUENTE] [NUEVO]** → renombra el archivo.
- **mv [options] [FUENTE1] [FUENTE2]... [DESTINO]**
- **mv [options] -t [DESTINO] [FUENTE1] [FUENTE2]...**

Si el último argumento nombra a un directorio existente, mv mueve cada uno de los otros ficheros a un fichero con el mismo nombre en ese directorio. Si no, si sólo se dan dos ficheros, renombra el primero al segundo. Es un error que el último argumento no sea un directorio y se den más de dos ficheros.

*mv directorioArchivos / \*. -> mueve todos los archivos en el subdirectorio indicado*  
*mv main.c main.java -> renombra el archivo main.c a main.java*

#### Comandos para modificar los atributos de los archivos:

- **chmod [options] [PERMISOS]... [ARCHIVO]** → Modifica los permisos.
- **chmod [options] [PERMISOSOCIAL] [ARCHIVO]** → Modifica los permisos usando una combinación numérica (777, todos los permisos).

Se puede usar el comando de forma directa, o indicando los permisos en forma octal:

**r** → lectura  
**x** → ejecución  
**w** → escribir

*Por ejemplo, para indicar lo siguiente:*

*El usuario puede leer, escribir y ejecutar el archivo.*  
*Los miembros del grupo pueden leer y ejecutar el archivo.*  
*Otros usuarios solo pueden leer el archivo.*

*chmod u=rwx,g=rx,o=r archivo.txt*

- **touch [options] [FICHERO]** → cambia las marcas de tiempo del archivo, pero si el archivo (cuyo nombre se pasa como argumento) no existe, entonces la herramienta lo crea.

#### Comandos para consultar algunos de sus atributos:

**file [options] [FILE]...** → Determina el tipo de archivo.



## Acceso a la información del SO relativa a sistemas de archivos

Los dos archivos fundamentales para obtener información de los sistemas de archivos son: **/etc/fstab** y **/etc/mtab**.

La información que muestra el archivo **/etc/fstab** es muy útil para comprender las opciones de montajes que se han realizado para cada uno de los sistemas de archivos que tenemos accesibles en nuestro sistema. Opciones:

- Modo de acceso a los archivos del sistema de archivos: **{rw|ro}**, lectura/escritura o solo lectura.
- Modo de acceso SUID: **{suid|nosuid}**, si/no.
- Montaje automático: **{auto|noauto}**, se permite o no el montaje automático. En el caso de no permitirlo no se realizará el montaje ni utilizando la orden *mount -a*.
- Ejecución de archivos: **{exec|noexec}**, si/no.
- Cuotas de usuario y de grupo: **usrquota, grpquota**.
- Valores por defecto de montaje (defaults): **rw, suid, dev, exec, auto, nouser, async**.
- Permitir a los usuarios montar un sistema de ficheros: **user, users, owner**.
- Propietario y grupo propietario de los ficheros del SA: **uid:500, gid=100**.
- Máscara a aplicar en los permisos de los archivos de nueva creación: **umask=022**.

Otro directorio del FHS que nos va a ser útil para obtener información es el **/proc**, el cual soporta el sistema de archivos virtual proc. Este directorio contiene archivos de texto que permiten acceder a información del estado del sistema.

Archivos con información de sistema de archivos que proporciona /proc:

- **/proc/filesystems** → enumera, uno por línea, todos los tipos de sistemas de archivos disponibles.
- **/proc/mounts** → sistemas de archivos montados actualmente, incluyendo los que se hayan montado manual o automáticamente tras el arranque del sistema.

## Actividades Sesión 1:

**Ejercicio: Visualiza el archivo /etc/passwd e indica cual es el formato de cada línea de dicho archivo. Para ello también puedes consultar el man o info de Linux. ¿Quién es el propietario de este archivo y cuáles son sus permisos?**

```
cat /etc/passwd
```

```
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
prueba1x500:500::/home/Prueba1:/bin/bash
```

Son 7 campos separados por “:” :

- 1 → User (login de la cuenta)
- 2 → x : clave de usuario encriptada (x = existe un archivo gshadow).
- 3 → UID

- 4 → GID
- 5 → Comentarios
- 6 → Directorio de trabajo del User
- 7 → Intérprete de comandos del usuario

**\*Nota:** Para el campo 5: **usermod -c** [comentario] [usuario]

**Ejercicio: Visualiza el archivo /etc/shadow desde un usuario distinto al root ¿Te da algún problema?**

**¿Sabes por qué? Intenta averiguarlo.**

Sin ser root no tenemos permiso para visualizar el archivo, ya que guarda información confidencial de los usuarios como las contraseñas.

Desde root se visualiza lo siguiente:

Prueba2:Pass2:18172:0:99999:7:::

Los campos son los siguientes, separados por : (dos puntos):

- 1.- Nombre del Usuario (Login)
- 2.- Password del login
- 3.- Días transcurridos desde 1/1/1970 donde el password se cambio por ultima vez
- 4.- mínimo número de días entre cambios de contraseña
- 5.- días máximos de validez de la cuenta
- 6.- Días que avisa antes de caducar la contraseña
- 7.- Días después de que un password caduque para deshabilitar la cuenta
- 8.- Fecha de caducidad desde 1/1/1970 donde la cuenta es deshabilitada y no podrá iniciarse sesión

**Ejercicio: 1. Crea un par de grupos y asignáelos a algunos de los usuarios de tu sistema.**

**2. ¿Qué información devuelve la orden id si estás conectado como root?**

- 1 ) Crear grupo → `groupadd nombre_grupo`  
Asignar a usuario → `gpasswd -a usuario nombre_grupo`  
Ver lo que se ha creado → `cat /etc/group`
- 2) `[root@localhost home]# id`  
`uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)`  
`[root@localhost home]# id Prueba1`  
`uid=501(Prueba1) gid=501(Prueba1) groups=501(Prueba1),503(Actividad)`

**Ejercicio: Utilizando la orden (find) que ya conoces para la búsqueda de archivos en el sistema de archivos, anota el nombre absoluto del archivo del kernel de Linux que se ha cargado en el sistema operativo que estás usando en el laboratorio de prácticas para acceso modo root.**

`find / -name Fedora`

Salida > `/usr/share/icons/Fedora`

**Ejercicio: Un programa que se ejecuta en modo root, ¿dónde podría guardar la información temporal de forma que ésta se mantuviese entre arranques del sistema?**

En el directorio /sbin. El directorio /sbin contiene típicamente ficheros indispensables para el arranque del sistema, además de los binarios contenidos en /bin.

**Ejercicio: INFORMACIÓN DE LOS SAs. Los archivos `/etc/fstab` y `/etc/mtab` muestran información sobre los sistemas de archivos que se encuentran montados en el sistema. ¿Cuál es la diferencia entre la información que muestra cada uno de ellos?**

`/etc/fstab` Es una lista creada por el usuario, que contiene la lista de volúmenes para ser montados por `mount` en el momento del arranque. Aquellas particiones que se deseen montar una vez arranque el computador deben ser colocadas en dicha lista.

`/etc/mtab` Es una lista creada por el sistema. Contiene los dispositivos actualmente montados. Si está conectado un disco duro, pero no montado, no aparecerá en dicha lista. Una vez sea montado, aparecerá.

**Ejercicio: Edita el archivo `/etc/fstab` del sistema de archivos que estás utilizando en modo root y anota y describe la información que tiene registrada. Si no conoces alguna opción puedes consultar el manual en línea: `man fstab`.**

**[Hacerlo iniciando con FEDORA]**

`[root@localhost ~]# cat /etc/fstab`

```
last login: Wed Oct 14 08:23:45 on tty0
[root@localhost ~]# cat /etc/fstab
#
# /etc/fstab
#
LABEL=ROOT                               /          auto       noatime1 1
tmpfs                                     /dev/shm   tmpfs      defaults0 0
tmp                                       /tmp       tmpfs      rw,mode=1777,fscontext=system_u:object_r:tmp_t:s0 0 0
devpts                                   /dev/pts   devpts     gid=5,mode=620 0 0
sysfs                                   /sys       sysfs      defaults0 0
proc                                    /proc      proc       defaults0 0
```

+ **Primera:** en este campo se indica el dispositivo o la partición donde se encuentra el filesystem.

+ **Segunda:** aquí va el punto de montaje para el dispositivo especificado.

+ **Tercera:** el tipo de sistema de archivos. Puede tomar varios valores, entre los que se destacan: `ext2`, `ext3`, `ext4`, `iso9660`, `nfs`, `ntfs`, `reiserfs`, `smbfs`, `swap`, `vfat`, `xf`.

+ **Cuarta:** en esta columna van las opciones para el montaje del filesystem.

+ **Quinta:** esta columna indica a la utilidad `dump` si debe o no hacer backup del filesystem. Puede tomar dos valores: 0 y 1. Con 0 se indica que no se debe backupear, con 1 que sí se haga. Lógicamente, depende de que se tenga instalado y configurado `dump`, por lo que en la mayoría de los casos este campo es 0

+ **Sexta:** en este caso se trata de una indicación para el `fsck` (comando que chequea el filesystem) y nuevamente se define con un valor numérico. Las posibilidades son 0, 1 y 2. El 0 indica que el filesystem no debe ser chequeado, mientras que el 1 y el 2 le dicen a `fsck` que sí lo chequee. La diferencia es que el 1 representa una prioridad mayor que el 2, por lo que debe utilizarse para el sistema raíz y el 2 para el resto de los sistemas de archivos.

**Ejercicio: Compara la información que contienen los cuatro archivos de texto que se han presentado en este apartado (`/etc/fstab`, `/etc/mtab`, `/proc/filesystems` y `/proc/mounts`). Describe en un párrafo para qué te sirve la información que registra cada archivo.**

`/etc/fstab` Es una lista creada por el usuario, que contiene la lista de volúmenes para ser montados por `mount` en el momento del arranque.

`/etc/mtab` Es una lista creada por el sistema. Contiene los dispositivos actualmente montados (incluso los que se montaron después, como un HDD)

`/proc/filesystems` Lista los sistemas de archivos que están soportados por el kernel.

`/proc/mounts` Aquellos sistemas de archivos que se montaron tras arrancar el sistema, o los que se montaron manualmente después.

## Repaso Sesión 1:

1. ¿Cómo se crearía un usuario con la opción de que se cree su directorio home? → `useradd -m`
2. ¿Qué fichero de configuración habría que modificar para, sin añadirle el parámetro a la orden `useradd`, se cree el directorio? → `/etc/login.defs` [Descomentar `CREATE_HOME` yes]
3. ¿Cómo podríamos borrar un usuario eliminando toda la información, directorio, etc. del sistema? → `userdel -r usuario`
4. ¿En qué fichero se puede ver la configuración que tenemos sobre el terminal bash? → `.bashrc`
5. ¿Cómo puedo ver la información de ‘envejecimiento’ de mi cuenta (tiempo de expiración de la contraseña, última vez que se cambió, etc)? → `cat /etc/shadow`

## Sesión 2. Herramientas de administración del SA

### **PARTICIÓN DE DISPOSITIVOS DE ALMACENAMIENTO SECUNDARIO**

Se considera almacenamiento secundario a aquella memoria conectada externamente (drive). Para poder usarlas en un Sistema Operativo es necesario establecer secciones (particiones) dentro del dispositivo físico que sean identificables y además permitan alojar un Sistema de Archivos correcto. A este proceso se le denomina “Partición de Disco”.

Una partición está constituida por un conjunto de sectores que formarán lo denominado Disco Lógico.

**\*Nota:** Sector. Es la unidad mínima de almacenamiento de información de un dispositivo de almacenamiento secundario. El tamaño varía, siendo los más comunes 512, 1024, 2048, 4096 bytes.

Cuando se crea una partición, será necesario asociarle una etiqueta que establezca que tipo de Sistema de Archivos se aplicará cuando se realice el formateo de la unidad. Esta información se almacena mediante un código numérico que determina el tipo de partición. Cada SO tiene sus propios códigos. En LINUX:

- **ext2** → **0x83**
- **swap** → **0x82**

`/sbin/sfdisk -T` → lista de los tipos de particiones soportados y sus códigos asociados.

Una partición de disco que a su vez puede dividirse se denomina Partición Lógica.

La partición primaria que se usa para alojar las particiones lógicas se conoce como partición extendida y tiene su propio tipo de partición: 0x05.

## **¿Cuántas particiones hago en mi dispositivo?**

Un dispositivo de arranque va a ser el dispositivo que utilice en primer lugar la BIOS de nuestra arquitectura para cargar en memoria el programa SO (¡ o un programa cargador de Sos!).

Si queremos que nuestro SO arranque desde el dispositivo sobre el que vamos a realizar las particiones necesitamos establecer la siguiente configuración de particiones:

- Una partición primaria.
- Una o más particiones swap.
- Ninguna, o las que quieras, partición(es), primaria(s) y lógica(s). Por supuesto que el número de particiones de esté dentro de los límites que establece la *compatibilidad hacia atrás* y el SO.

La configuración de particiones de cualquier dispositivo de almacenamiento secundario que no se vaya a utilizar como dispositivo de arranque sino simplemente para almacenar información, será la siguiente:

- Una o más particiones primarias o lógicas (dentro de los límites).
- Ninguna o las que quieras partición(es) swap.

## **¿Qué directorios de primer nivel de estándar FHS deberían estar soportados por una partición independiente?**

Toda la estructura del directorio FHS puede estar soportada por una única partición (la partición de arranque etiquetada como "/"). Es conveniente en determinadas situaciones establecer particiones independientes que soportan la información que se almacene en determinados directorios de la estructura.

- **/home** → se almacenan los directorios de inicio de todos los usuarios con cuenta en el sistema. De cara a la instalación de versiones nuevas de Linux, si el /home está soportado en una partición independiente no tendremos problemas con nuestra partición raíz y podremos formatearla e instalar en ella la nueva versión del Sistema Operativo.

- **/usr** → almacena la mayoría de los ejecutables binarios del sistema, así como sus demás archivos adicionales, incluyendo los de documentación. Además si instalamos el paquete con el árbol de las fuentes del kernel también va a parar aquí.

\* **Nota:** El **spooling** es útil en caso de dispositivos que acceden a los datos a distintas velocidades. O en aquellos casos en que no hay comunicación directa entre los programas que escriben los datos y aquellos que los leen.

## **Asignación de un sistema de archivos a una partición (formateo lógico)**

Una vez las particiones se han realizado y están disponibles, se debe proceder a asignar el sistemas de Archivos adecuado a cada una de las particiones. A parte de las dedicadas al intercambio (swap), se utilizan tres Sistemas de Archivos:

- **ext2** → SA de alto rendimiento de Linux. Usado para discos duros, memorias flash y medios extraíbles. Ofrece mejor rendimiento en términos de velocidad de transferencia de E/S y uso de CPU.

- **ext3** → Incluye un "registro por diario" (journaling).

\**Journaling*. Es un mecanismo por el cual un sistema informático puede implementar transacciones. Almacena la información necesario para restablecer los datos afectados por la transacción. Permite evitar la corrupción de las estructuras de datos que soportan la información del SA.

- **ext4** → Es el estándar actual usado por Linux. Tiene una Estructura similar a las del ext3, que presenta las siguientes mejoras:

-*Extensiones*. Permite describir un conjunto de bloques de disco contiguos (mejora rendimiento de E/S con archivos de gran tamaño. Reduce fragmentación de disco)

-Asignación retardada de espacio en disco (*allocate-on-flush*). Consiste en retrasar la reserva de bloques de memoria hasta que la información está a punto de ser escrita en el disco, a diferencia de otros sistemas de archivos, los cuales reservan los bloques necesarios antes de ese paso. Esto mejora el rendimiento y reduce la fragmentación al mejorar las decisiones de reserva de memoria basada en el tamaño real del fichero.

### Ajustes de parámetros configurables de un SA y comprobación de errores

Una vez que están disponibles nuestros nuevos Sistemas de Archivos se puede usar la orden **tune2fs** → ajusta parámetros de los SA ext2/ ext3 /ext4.

#### **Tune2fs [options] [DISPOSITIVO]**

- **l** → se muestra por pantalla el contenido del superbloque del SA.

- **c [max\_mount\_counts]** → establece el número máximo de montajes que se puede llegar a relaizar sin realizar una comprobación de la consistencia del SA.

- **L [ETIQUETA]** → poner un etiqueta al SA.

- **r [reserved\_blocks\_count]** → establece el número de bloques reservados del sistema de archivos.

- **u [user]** → establece el suario el cual puede hacer uso del bloque reservado del Sistema de Archivos. [user] puede ser un número UID o un nombre. (si se da un nombre de usuario este se convierte en un UID antes de hacer la reserva exclusiva).

Pueden ocurrir situaciones en las que sea necesario que el administrados ejecute manualmente las comprobaciones y repare las posibles inconsistencias. Linux proporciona la **fsck** para llevar a cabo esta labor. Algunas de las situaciones de inconsistencia de metadatos del SA:

- Bloques que están asignados simultaneamente a varios archivos.
- Bloques marcados como libres pero que están asignados a un archivo determinado.
- Bloques marcados como asignados a un archivo determinado pero que realmente están libres.
- Inconsistencia del número de enlaces a un determinado archivo.
- i-nodos marcados como ocupados pero que realmente no están asociados a ningún archivo.

## Montaje y desmontaje de Sistemas de Archivos

Una vez se dispone de los nuevos Sistemas de Archivos, solo falta ponerlos a disposición del usuario, es decir, hacer que sean accesibles dentro de la jerarquía de directorios (es decir, poder navegar entre ellos).

Hasta este momento solamente disponemos de los sistemas de archivos creados en las particiones correspondientes.

Para poder crear archivos y directorios en estos sistemas de archivos es necesario solamente indicar en qué directorio se montará (aquí es donde crecerá la nueva rama del espacio de nombres), y cuál es el nombre del archivo especial de dispositivo que representa la partición en donde reside el sistema de archivos.

*Punto de montaje* → directorio que se utiliza como punto de partida para el sistema de archivos montado.

***mount [OPCIONESMOUNT] [FICHERO] [DESTINOMONTAJE] [PERMISOSFICHERO]***

(mirar apuntes josu antes de los ejercicio 2.4 y 2.5)

## Administración de software

Entre las responsabilidades de un administrador de un sistema operativo están las tareas de instalar nuevo software de aplicación que los usuarios necesitan ejecutar, o más bien mantener la seguridad del sistema mediante actualizaciones. Estas tareas se han de llevar a cabo como root.

Otras tareas como compilación de código fuente y construcción de código ejecutable lo puede llevar a cabo un usuario normal.

*Paquete Software* → es un archivo que contiene un conjunto de programas complementarios, donde la mayoría dependen de otros. Dichas especificaciones de dependencias están incluidas en el propio paquete como metadatos. El software instalable en Linux se estructura en paquetes.

*Gestores de paquetes* → herramientas gráficas que se usan para la instalación de paquetes.

### **Gestores de paquetes**

Es la forma más simple de instalar y actualizar software de un sistema operativo. Habitualmente son proporcionados por la distribución del propio Sistema Operativo.

Los paquetes se identifican dependiendo del Sistema Operativo, los de Linux basados en Debian tienen la extensión “.deb”.

Los gestores de paquetes se pueden instalar en modo línea de órdenes y se dividen en dos niveles:

- *Alto nivel* → gestores de paquetes apt-get y YUM, sobre los que a su vez se suelen proporcionar interfaces gráficos, como por ejemplo PackageKit y gnome-packagekit sobre YUM. Suelen ser más interactivos y pueden realizar búsquedas y actualizaciones automáticas.

- *Bajo nivel* → como dpkg y rpm. Estos pueden llegar a ser más precisos y potentes

## **APT y YUM**

**APT** (*Advances Packaging Tool*) fue originalmente desarrollado por Debian Linux y modificado para su uso también como paquetes RPM.

Tanto APT, como su interfaz gráfico Synaptic, son fáciles de utilizar. Existen varios proveedores principales de paquetes y repertorios para APT, pero no se deben utilizar simultáneamente porque a veces existen conflictos entre versiones.

Al parecer, **YUM** es mejor herramienta para gestionar RPM, pues al parecer APT contiene más código innecesario que se utiliza realmente para los paquetes .deb.

Puede usarse yum --help para una información más precisa sobre el gestor YUM. Los más habituales:

- **yum list** → Lista los paquetes disponibles en los repositorios para su instalación.

- **yum list installed** → Paquetes actualmente instalados.

- **yum list updates** → Muestra paquetes con actualizaciones disponibles.

- **yum install [PAQUETE]** → Instala el paquete especificado.

- **yum update** → Actualiza todos los paquetes actualizados.

- **yum remove [PAQUETE]** → Elimina el paquete especificado.

## **RPM**

Comprueba dependencias e incluye opciones como la verificación de la revisión y las firmas de seguridad de privacidad GNU que permiten que los paquetes se distribuyan con la seguridad (libre de virus).

**rpm [options] [NOMBREPAQUETES]...**

Opciones comunes:

- Instalación de nuevos paquetes

**rpm -i [NOMBREARCHIVOPAQUETE]** → Si la operación tiene éxito no se mostrará ningún mensaje.

- Eliminación de paquetes instalados

**rpm -e [NOMBREPAQUETE]** → Si la operación tiene éxito no se mostrará ningún mensaje

- Actualización de Paquetes Instalados



***rpm -U [NOMBREARCHIVOPAQUETE]*** → Se actualiza descargando el paquete de la nueva versión y elimina la anterior.

***rpm -F [NOMBRESERVIDOR-HTTP/FTP]*** → Se busca el paquete en el servidor indicado y se prepara la actualización.

- Obtención de Información sobre Paquetes Software

***rpm -qa | grep [PARTENOMBREPAQUETEBUSCADO] | sort*** → Busca paquetes instalados bien por su nombre o parte de este.

***rpm -qi [NOMBREPAQUETE]*** → Información precisa del paquete instalado indicado.

- Verificación e Integridad de la Instalación

***rpm -V [NOMBREPAQUETE]*** → Consulta en la base de datos para verificar la instalación de un paquete. Si la operación es de éxito, no muestra ningún mensaje.

## **Administración de cuotas**

Mediante un sistema de Cuotas es posible limitar el uso del disco por parte de usuarios de una forma más flexible.

Las cuotas permiten limitar el número de recursos de un Sistema de Archivos usado por el usuario. Estos son los bloques de disco y los i-nodos.

**\*Nota: i-nodo** → es el registro en el disco, el cual contiene información sobre el archivo carpeta como puede ser su tamaño, los propietarios, etc; SALVO el contenido del mismo y su nombre. Se identifica por un entero único.

El número de i-nodos en una cuenta equivale al número de archivos y carpetas almacenadas en ella.

Si se quiere trabajar con el Sistema de Cuotas será necesaria la instalación del paquete quota (no instalado por defecto).

El sistema de cuotas establece dos límites para restringir el uso de bloques e i-nodos. Estos límites se pueden establecer para usuarios y/o grupos y para bloques y/o i-nodos.

- **Límite Hard** → define el límite definitivo del espacio asignado del disco duro que no debe sobrepasarse. Cuando un usuario alcanza el límite, se le impide continuar usando el disco duro (no podrá ampliarse el tamaño de los archivos ni usar más i-nodos, es decir, crear nuevos archivos). En este caso, puede solicitar una ampliación del espacio asignado al administrador o liberar espacio borrando datos.

- **Límite Soft** → es un límite de advertencia cercano al límite efectivo pero que deja algo de margen para ser sobrepasado. Cuando se sobrepasa se avisa al usuario, considerándose “*over quota*”. El usuario podrá hacer uso de la capa del disco hasta alcanzar el límite definitivo. A este periodo se le denomina “Periodo de Gracia” y por lo general, lo establecen los administradores.

## Crear un archivo de cuotas

Después de volver a montar cada sistema de archivos con cuotas, el sistema puede funcionar con cuotas de disco.

Sin embargo, el sistema de archivos mismo no está listo para soportar cuotas. El próximo paso es ejecutar el comando **quotacheck**.

El comando quotacheck examina los sistemas de archivos con cuotas activadas y construye una tabla del uso del disco por sistema de archivo. La tabla es luego usada para actualizar la copia del uso del disco del sistema operativo. Además, los archivos de cuotas de disco del sistema de archivos, son actualizados.

Para crear los archivos de cuotas (**aquota.user** y **aquota.group**) en el sistema de archivos, use la opción **-c**.

*Por ejemplo, si las cuotas del usuario y grupos están activadas para la partición /home, se crearán los archivos en el directorio /home:*

*quotacheck -acug /home*

*quotacheck [options] [DIRECTORIOMONTAJE]*

## Actividades Sesión 2:

Ejercicio:

## Repaso Sesión 2:

1. ¿Cómo podemos ver la lista de tipos de particiones con los que nos permite trabajar nuestro SO?  
→ /sbin/sfdisk -T
2. ¿Cómo podemos ver cuál es el dispositivo que acabamos de insertar en el sistema? → dfisk -l
3. ¿Cómo podemos ver la lista de dispositivos que están montados? → /proc/mounts
4. ¿Cómo podríamos saber los dispositivos loop que están usándose? → losetup -a o losetup -l
5. Supongamos que no sabemos qué dispositivo loop está libre ¿cómo podríamos asignar un fichero al primer dispositivo loop que estuviera libre? → losetup -f

## Sesión 3. Monitorización del sistema

### Control y gestión de la CPU

#### Orden UPTIME

Muestra la siguiente información:

- Hora actualiza.
- Tiempo que lleva en marcha el sistema.
- Número de usuarios conectados.
- Carga media del sistema en los últimos 1, 5 y 15 minutos.

Ejecutando **w** obtenemos la misma información, junto a lo que hacen los usuarios conectados.

**\* Nota:** *Carga del sistema* → se entiende por esto el número de procesos en la cola de ejecución del núcleo. Un sistema con un comportamiento normal no debe superar una carga de valor 1, aunque en sistema multicore habrá que multiplicar ese número por el número de cores.

#### Orden TIME

Mide el tiempo de ejecución de un programa y muestra el resumen del uso de los recursos del sistema. (También puede usarse para temporizar una orden sencilla).

***time [options] [ORDEN] [ARGUMENTOS]...***

El comando ejecuta el programa ORDEN con los argumentos suministrados.

***Ejemplo: orden ps***

Primero ejecuta el programa ps, y después nos muestra los valores de los tiempos *real* (tiempo total transcurrido desde que ha invocado el comando), *user* (cantidad de tiempo actualmente consumido en la CPU en modo usuario) y *sys* (tiempo que ha ejecutado en modo supervisor). → se pueden visualizar los procesos en el sistema, y obtener información de los mismos.

Se puede indicar la siguiente fórmula que determina cómo calcular el tiempo de espera de un proceso:

$$\underline{T_{espera} = real - user - sys}$$

#### Orden NICE y RENICE

Linux realiza una planificación por prioridades, donde, por defecto, un proceso hereda la prioridad del padre.

Con el comando *NICE* se puede modificar el valor por defecto de un proceso de su prioridad. El rango va de [-19,20] siendo un valor negativo el que aumenta la posibilidad de ser ejecutado dicho proceso (¡Solo root puede dar un valor negativo!).

***nice* -[VALOR] [PROCESO]**

**Ejemplo:** *nice -5 konqueror* → **aumenta en 5 el valor de prioridad de la ejecución de konqueror.**

*nice -- konqueror* → **decrementa en 10 el valor** de la prioridad.

***nice -n[numero] [PROCESO]*** → Damos el valor ‘número’ de prioridad al proceso.

El comando *RENICE* permite alterar el valor de prioridad de uno o más procesos en ejecución.

Estructura: ***renice* [VALOR] [PID]**

## **Orden PSTREE**

Visualiza un árbol de procesos en ejecución. Se trata de un programa que muestra los procesos, desde la línea de comandos, en forma de diagrama de árbol.

Difiere de ‘ps’ en que ps muestra los procesos en una lista en lugar de un diagrama, pero proporciona información más detallada sobre estos.

El programa pstree facilita información sobre la finalización de una serie de procesos relacionados entre sí, esto es, todos los descendientes de un proceso particular. El programa deja claro desde un principio que proceso es el primario y cuales son los secundarios. Esto evita buscar las relaciones entre los diversos procesos de manera manual.

Opciones:

**-a** → Muestra los argumentos de la línea de órdenes.

**-A** → Usa caracteres ASCII para dibujar el árbol.

**-G** → Usa los caracteres de VT100.

**-h** → Resaltar el proceso actual y sus antepasados.

**-H** → Igual que -h pero para que el proceso se especifique.

**-l** → Usa un formato largo, por defecto las líneas se truncan.

**-n** → Ordena los procesos por el PID de su antecesor en vez de por el nombre (ordenación numérica).

**-p** → Desactiva el mostrar los PIDs entre paréntesis después del nombre de cada proceso.

**-u** → Si el uid de un proceso difiere del de su padre, el nuevo se pone entre paréntesis después del nombre del proceso.

- V** → Visualiza información sobre la versión.
- Z** → Muestra el contexto de seguridad para cada proceso.

## **Orden PS**

Muestra información sobre los procesos en ejecución. Suele usarse con las opciones -ef.

- e** → Selecciona todo proceso que esté en el sistema.
- f** → Muestra la información completa (con -l se muestra más información).

Usarlo sin argumentos muestra solo aquellos procesos lanzados por el usuario que ejecuta el comando.

Usando el comando con los argumentos *axu*, muestra la siguiente información:

***ps axu***

- USER** → Usuario que lanzó el programa/proceso.
- PID** → Identificador único del proceso.
- PPID** → Identificador del proceso padre.
- %CPU** → % entre el tiempo usado realmente y el que lleva en ejecución.
- %MEM** → Fracción de memoria consumida (estimada).
- VSZ** → Tamaño virtual del proceso (codigo+datos+pila) en KB.
- RSS** → Memoria real usada.
- TTY** → Terminal asociado con el proceso.
- STAT** → Estado del proceso indicado mediante una letra (con su significado asociado).

- R** → en ejecución o listo (Running o Ready).
- S** → durmiendo.
- T** → parado.
- Z** → proceso zombie.
- D** → durmiendo ininterrumpible (normalmente E/S).
- N** → prioridad baja (>0).
- <** → prioridad alta (<0).
- s** → líder de sesión.
- l** → tiene multi-thread.
- +** → proceso foreground
- L** → páginas bloqueadas en memoria.

## **Orden TOP**

Muestra información continua de la actividad del procesador en tiempo real. Muestra las tareas que más uso hacen de CPU, y además tiene una interfaz interactiva para manipular los procesos. Los procesos son ordenados de forma DECRECIENTE en base al uso de CPU. Dicha lista se actualiza de forma interactiva cada 5 sec. aprox.

Las cinco primeras líneas que muestra son información general del sistema.

- Las estadísticas que mostraría la orden uptime.

- Estadísticas sobre los procesos del sistema.
- Estado actual de la CPU (porcentaje en uso por usuarios, por el sistema, por procesos con valor nice positivo, por procesos esperando E/S, CPU desocupada, tratando interrupciones hardware o software, en espera involuntaria por virtualización).
- Memoria total disponibles, usado, libre, cantidad usada en buffers y memoria caché de página.
- Swap total disponible, usado, libre.

Los datos de la parte inferior son similares a los del ps, excepto SHR que muestra la cantidad de memoria compartida usada por la tarea.

Opciones:

- r** → cambia la prioridad de algún proceso.
- k** → matar o enviar una señal.
- Ordenarlos con diferente criterio (**PID, con N; CPU con p; tiempo, con A**).
- n** → cambia el número de procesos que se muestran.
- q** → salir.

## **Orden MPSTAT**

Muestra las actividades de los procesadores (en caso de múltiples núcleos), contando desde cero para el primer núcleo. También nos proporciona un promedio de las actividades de todos los procesadores en conjunto.

Pueden usarse parámetros para definir la cantidad de tiempo entre cada toma de datos y el número de informes deseados.

**mpstat [INTERVALO] [NÚMERO]**

+ **[INTERVALO]** → cada cuantos segundos debe mostrar los datos.

+ **[NÚMERO]** → cuantos muestreos se solicitan.

**\*\* Necesario tener instalado SYSSTAT.**

Información mostrada:

**CPU** : número del procesador

**%user** : porcentaje de uso de la CPU con tareas a nivel de usuario.

**%nice** : porcentaje de uso de la CPU con tareas a nivel de usuario con prioridad “nice” (> 0)

**%sys** : porcentaje de uso de la CPU para tareas del sistema (no incluye el tratamiento de interrupciones) (modo núcleo).

**%iowait** : porcentaje de tiempo que la CPU estaba “desocupada” mientras que el sistema tenía pendientes peticiones de E/S.

**%irq** : porcentaje de tiempo que la CPU gasta con interrupciones hardware.

**%soft** : porcentaje de tiempo que la CPU gasta con interrupciones software (la mayoría son llamadas al sistema).

**%idle** : porcentaje de tiempo que la CPU estaba “desocupada” y el sistema no tiene peticiones de disco pendientes.

*intr/s* : número de interrupciones por segundo recibidas por el procesador.

## Control y gestión de memoria

### Orden FREE

Consume menos recursos (CPU y memoria) que top. Se usa para visualizar el uso actual de memoria. Informa del consumo de:

- Memoria Real o Principi (RAM) instalada en la computadora.
- Memoria de espacio de intercambio (swap)

*\*Nota:* los campos buffers y cached, la memoria que indican son memoria libre.

### Orden VMSTAT

Sirve para supervisar el sistema. Permite obtener un detalle general de los procesos, E/S, uso de memoria/swap, estado del sistema y actividad de la CPU. En esencial para entender que está pasando en tu sistema, detectar cuellos de botella, etc.

Su primera salida proporciona una media desde el último arranque del sistema operativo y se pueden obtener informes sobre el uso durante el periodo de tiempo actual, indicando el periodo de tiempo en segundos y número de iteraciones deseadas.

Si se quiere guardar la información en un archivo se podrá usar de la forma:

**vmstat [options] [RETRASO] [CONTADOR] > [ARCHIVO]**

**\$ vmstat**

```
procs -----memoria----- --swap-- ----io---- -sistema-- -----cpu-----
r b swpd libre búfer caché si so bi bo in cs us sy id wa st
0 0 0 1793444 425496 3534464 0 0 27 50 213 76 6 2 91 1 0
```

Información que indica cada columna

Información sobre los procesos, procs

- **r** --> Número de procesos ejecutables (en cola de ejecución)
- **b** --> Número de procesos en estado "dormido"

Información sobre la memoria, memory

- **swpd** --> Muestra la cantidad de memoria libre de la swap (de intercambio)
- **libre (free)** --> Cantidad de memoria libre
- **buffer** --> Cantidad de memoria en el buffer (las memorias intermedias)
- **cache** --> Memoria caché disponible

Información respecto la memoria swap

- **si** --> Memoria intercambiada DESDE el disco
- **so** --> Memoria intercambiada HACIA el disco

#### Información sobre entrada salida, E/S, IO

- **bi** --> Bloques recibidos de un dispositivo de bloques
- **bo** --> Bloques enviados de un dispositivo de bloques

#### Información respecto el sistema

- **in** --> Indica el número de interrupciones por segundo
- **cs** --> Número de cambios de contexto por segundo

#### Información sobre la CPU

- **us** --> Muestra el porcentaje de uso por usuario
- **sy** --> Muestra el porcentaje de uso a nivel de sistema
- **id** --> Porcentaje de tiempo que la CPU está desocupada
- **wa** --> Muestra las esperas de Entrada y Salida
- **st** --> Tiempo robado de una máquina virtual

Las E/S de disco suelen ser un cuello de botella para la mayoría de los sistemas, así cualquier aspecto que impacte en el acceso al disco puede provocar importantes diferencias en el sistema.

Por ejemplo, una alta demanda de memoria puede provocar en el sistema utilice intensivamente el espacio de intercambio transfiriendo continuamente información de memoria a disco y viceversa.

## **Control y gestión de dispositivos E/S**

El SO necesita mantener una estructura de datos por cada archivo que contenga la información que le es necesario mantener para poder trabajar con él. (Metadatos de archivo, o Atributos de archivo).

Un metadato de archivo fundamental es el "Nombre de archivo". Es usado para identificar la información que contiene y ubicarlo en un directorio concreto.

En UNIX se almacena una referencia a los metadatos en la entrada del directorio (los **inodos**, o **inode**)

Existen unos tipos de archivos especiales de dispositivo que soportan los SA tipo UNIX, archivos especiales para dispositivos de bloques y para dispositivos de caracteres, los cuales proporcionan al usuario una interfaz abstracta para el trabajo con dispositivos de E/S y almacenamiento secundario. Para crear estos archivos se usa el comando MKNOD (más adelante)

Los datos de los archivos como los metadatos de archivo deben almacenarse en los dispositivos de almacenamiento persistente.

Para la gestión de este almacenamiento, los sistemas de archivos tipo UNIX mantienen, entre sus metadatos de SA, información relativa a bloques de disco libres/ocupados e inodos libres/asignados. Para acceder a esta información utilizaremos las órdenes df y du.



## Consulta de información de archivos

Por general, si se usa la función `ls` sin indicar un directorio, muestra la información del directorio de trabajo. Si además se usa la opción `-l` al usar este comando, se muestra información relativa a los metadatos de los archivos de dicho directorio.

**`ls [options] [ARCHIVO/DIRECTORIO]`**

- **`l`** → Escribe los permisos del fichero, número de enlaces que tiene, nombre del propietario, el de su grupo, el tamaño, una marca de tiempo, el nombre del fichero. (El formato largo). Por defecto muestra la fecha de modificación.

- **`n`** → Lista los UID y GID numéricos en vez de los nombres.

- **`a`** → No ignora los directorios que empiecen por "." (entradas ocultas).

- **`i`** → Añade el número del inodo.

- **`h`** → Añade una letra indicativa de tamaño, tal como M para megabytes binarios.

```
$> ls -lai
total 20
6163598 drwxr-xr-x ...
...
```

Indica la información de metadatos de archivo, como el tipo de archivo y permisos. El primero de todos los caracteres indica lo siguiente:

- **`-`** → archivo regular.
- **`d`** → directorio.
- **`l`** → enlace simbólico. (Ya veremos más adelante que hay dos tipos de enlace en UNIX pero solo un tipo de archivo enlace).
- **`b`** → archivo especial de dispositivo de bloques.
- **`c`** → archivo especial de dispositivo de caracteres.
- **`p`** → archivo FIFO para comunicaciones entre procesos.

El comando `ls` además tiene la capacidad de ordenar la información listada a través de una serie de opciones de ordenación que `ls` permite:

Especialmente indicadas para '*long listing format*'.

- + **`ls -X`** → Ordena alfabéticamente por la extensión de entrada del directorio.
- + **`ls -t`** → Ordena por la fecha de modificación
- + **`ls -u`** → Ordena por la fecha de acceso

+ **ls -c** → Ordena por el campo *ctime* (fecha de la última modificación de la información del estado del fichero).

## Consulta de metadatos del SA

Para poder asignar espacio en disco, mantiene además de otros metadatos, información relativa a bloques de disco libres/ocupados e inodos libres/asignados. Es una información muy importante ya que la falta de alguno de estos dos recursos puede provocar degradación en el uso del SA por parte del usuario, e incluso llegar al caso de no poder volver a usarlo.

### **COMANDO DF**

Permite visualizar para cada SA montado información sobre

- La capacidad de almacenamiento total
- Espacio usado para almacenamiento
- Espacio libre restante
- Punto de montaje

```
[root@localhost ~]# df
```

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
LABEL=ROOT	1032088	411400	568260	42%	/
tmpfs	1032088	411400	568260	42%	/dev/shm
/tmp	509748	0	509748	0%	/tmp

Si al comando se añade la **opción -i** se visualizará información sobre los inodos de cada SA montado (Sistema de Archivos).

Si se usa la opción **-h** se verán los tamaños en un formato legible y entendible fácilmente para un humano

```
> df -h
```

Filesystem	Size	Used	Avail	Use%	Mounted on
LABEL=ROOT	1008M	451M	507M	48%	/
tmpfs	1008M	451M	507M	48%	/dev/shm
/tmp	499M	0	499M	0%	/tmp

Por defecto usará MEGABYTES, pero si se desea que se expresa en GIGABYTES, usar la **opción -BG**

```
> df -BG
```

Filesystem	1G-blocks	Used	Available	Use%	Mounted on
LABEL=ROOT	1G	1G	1G	48%	/
tmpfs	1G	1G	1G	48%	/dev/shm
/tmp	1G	0G	1G	0%	/tmp

Si con el comando **df**, se usa la **opción -T**, muestra información de una partición en particular

```
> df -hT /etc
```

Filesystem	Type	Size	Used	Avail	Use%	Mounted on
LABEL=ROOT	auto	1008M	451M	507M	48%	/

## COMANDO DU

Visualiza el espacio en disco que gasta un directorio, así como todo el subárbol que comienza en él. Proporciona el número de bloques de disco asignados a todos los archivos y directorios que cuelgan del directorio especificado

- La última línea muestra la cantidad total de bloques de disco usados por el subárbol
- Contabiliza el número de bloques e disco asignados ya estén o no ocupados en su totalidad  
Al espacio libre dentro de un bloque se le denomina "*Fragmentación interna*".

Una opción interesante para saber el tamaño total de un directorio en el disco es **-s**, que combinada con **-h** (un formato legible para humanos) lo hace más preciso a la hora de comparar.

## Creación de enlaces a archivos

El objetivo de los enlaces a archivos es disponer de más de un nombre para los archivos en nuestro espacio de nombres de archivo soportado por la estructura jerárquica de directorios.

*Los enlaces a archivos pueden considerarse como referencias a otros archivos, bien a su nombre, enlaces simbólicos, bien a sus metadatos, enlaces duros.*

**\*Nota:** Diferencia entre Enlaces Duros y Simbólicos.

**\*Enlaces Duros.** Para entender esto, en primer lugar hay que recordar que cada fichero y directorio tienen asignado un único número entero denominado **inodo**. La información de dicho inodo es la siguiente

- Permisos del archivo/directorio
- Propietario del archivo/directorio
- Posición/ubicación del archivo/directorio dentro del disco duro
- Fecha de creación del archivo/directorio
- etc (solo interesa esto)

Un enlace duro es un ARCHIVO que APUNTA al MISMO CONTENIDO almacenado en disco que el archivo original.

Los archivos originales y los enlaces duros comparten el MISMO INODO. Podría decirse que un enlace duro no es más que una forma de identificar un contenido almacenado en el disco duro con un nombre distinto al original.

Solo podrá crearse un enlace duro de un archivo que se encuentre en la misma partición, ya que: CADA PARTICIÓN POSEE SU PROPIA TABLA DE INODOS.

**\*Enlace Simbólicos.** De forma rápida, se puede entender como un acceso directo en Windows. En este caso, los enlaces simbólicos apuntan al nombre de un archivo, y entonces este archivo (ahora identificado) apunta al contenido en el disco duro.

Un enlace simbólico tiene su propio número de *inodo*, diferente al archivo al que apunta. Esto permite crear enlaces simbólicos de archivos/directorios que no estén en la misma partición.

Explicado lo anterior, se puede establecer que al crear un archivo, no importa el tipo, se establece un primer enlace duro sobre su inodo asociado en el momento en el que el Sistema Operativo crea la entrada en el directorio donde se ubicará.

Se puede comprobar usando el comando `ls -la` que todos los directorios poseen dos enlaces duros creados por el SO para poder mantener la jerarquía de directorios (`directorio_padre`, `directorio_hijo`).

Se encontrará el carácter "." para indicar la entrada del propio directorio y el carácter ".." para indicar la entrada del directorio padre.

Para crear enlaces duros o simbólicos se usará en comando `ln`

**`ln [options] [NOMBREARCHIVOQUESEENLAZA] [NUEVONOMBREDELARCHIVO]`**

**`ln -s [] []` → *SoftLink, o enlace Simbólico***

**`ln -P [] []` → *hardLink, o enlace Duro***

En la tabla que aparece al usar el comando `ls -lai`, la columna anterior al NAME del propietario del archivo se denomina contadores de enlaces. Mantiene el número de enlaces duros a archivos. El objetivo es poder liberar el inodo cuando todos los archivos que usen dicho inodo hayan sido eliminados.

\*Si se liberase el inodo con un contador de enlaces mayor que 0, se podría intentar acceder al archivo y se produciría una inconsistencia entre el espacio de nombres y los metadatos de archivo.

## **Archivos especiales de dispositivo**

Existen dos tipos principales de archivos especiales de dispositivo:

- **Bloques.** Normalmente coinciden con los dispositivos de almacenamiento persistente, los *ramdisks* y los dispositivos *loop*.

- **Caracteres.** Representan a puertos serie, paralelo y USB, consola virtual (console), audio, los dispositivos de terminal (tty\*), y muchos más.

*crw-rw---- 1 root disk 21, 0 2014-04-25 11:39 sg0*

Hace referencia a una interfaz del disco que realiza copias completas de la información almacenada. Útil para realizar copias de seguridad o duplicar el disco.

**`mknod --> mknod [options] [NOMBRE] [MAJOR] [MINOR]`**

Permite crear archivos especiales de dispositivo (bloque o caracteres → buffered o unbuffered).

Se indica el nombre del archivo y dos números identificados como major y minor que permiten identificar al dispositivo en el kernel.

- *Major, o número principal.* Determina el controlador al que está conectado.

- *Minor, o número secundario.* Determina el dispositivo en sí.

## **Actividades Sesión 3:**

**Ejercicio: Responde a las siguientes cuestiones y especifica, para cada una, la opción que has utilizado (para ello utiliza `man` y consulta las opciones de las órdenes anteriormente vistas:**

> uptime

06:21:30 up 0 min, 1 user, load average: 0.23, 0.09, 0.03

- a) ¿Cuánto tiempo lleva en marcha el sistema? → 0 min
- b) ¿Cuántos usuarios hay trabajando? → 1 usuario
- c) ¿Cuál es la carga media del sistema en los últimos 15 minutos? → 0.03

**Ejercicio: a) 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**  
**el valor de la variable es 1000**

SCRIPT prueba\_procesos:

```
#!/bin/bash
#$1 cogerá el segundo parámetro ya que $0 es el programa.
It=$1
sum=0
for (( c=0 ; c<it ; c++));
do
sum=$((sum+1)) #Para usar el valor de la variable se debe usar $ ; si quiere solo modificarse, no.
done
printf "La suma es $sum\n"
```

Para ejecutar este programa:

**chmod a+x prueba\_procesos.sh**  
**./prueba\_procesos.sh [número]**

**b) 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?**

Para ejecutar este programa en **background (segundo plano)** se hace uso del ampersan **&**:  
**./prueba\_procesos.sh [número] &**

Para comprobar la información de los procesos y su prioridad: **ps -l** ; donde la columna NI es la prioridad del proceso.

**c) Obtén los tiempos de finalización de cada uno de los guiones del apartado anterior.**

Para obtener los tiempos de finalización usaremos: **time ./prueba\_procesos.sh [número] %**

**Ejercicio: a) La orden pstree muestra el árbol de procesos que hay en ejecución. Comprueba que la jerarquía mostrada es correcta haciendo uso de la orden ps y de los valores "PID" y "PPID" de cada proceso.**

**ps -f**

**b) Ejecuta la orden ps con la opción -A, ¿qué significa que un proceso tenga un carácter "?" en**

### la columna etiquetada como TTY?

Significa que no tiene una terminal asociada

### Ejercicio: a) ¿Qué porcentaje de tiempo de CPU se ha usado para atender interrupciones hardware?

(No es posible instalar mpstat. Se ha tomado un ejemplo de Internet)

```
09:22:45 AM CPU  %user  %nice  %sys %iowait  %irq  %soft %steal  %idle  intr/s
09:22:45 AM all   3.58   0.09   0.94  0.26     0.01  0.05  0.00  95.07  24.42
```

%irq = 0.01

b) ¿Y qué porcentaje en tratar interrupciones software? → %soft = 0.05

c) ¿Cuánto espacio de swap está libre y cuánto ocupado?

Para saberlo usaremos la orden top:

> top

```
top - 10:11:25 up 3:50, 1 user, load average: 2.00, 1.58, 0.82
Tasks: 39 total, 3 running, 36 sleeping, 0 stopped, 0 zombie
Cpu(s): 1.0%us, 0.1%sy, 0.0%ni, 98.7%id, 0.2%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1020512k total, 40876k used, 979636k free, 5508k buffers
Swap: 0k total, 0k used, 0k free, 18564k cached
```

0 en ambos casos

### Ejercicio: Resuelve las siguientes cuestiones relacionadas con la consulta de metadatos del sistema de archivos:

1. Comprueba cuántos bloques de datos está usando la partición raíz del sistema UML del laboratorio. Ahora obtén la misma información pero expresada en “human readable format”: Megabytes o Gigabytes. Para ello consulta en detalle el manual en línea.

Se usará la opción -h con el comando df

> df -h

Filesystem	Size	Used	Avail	Use%	Mounted on
LABEL=ROOT	1008M	451M	507M	48%	/
tmpfs	1008M	451M	507M	48%	/dev/shm
/tmp	499M	0	499M	0%	/tmp

2. ¿Cuántos inodos se están usando en la partición raíz? ¿Cuántos nuevos archivos se podrían crear en esta partición?

Con el comando df y la opción -i se podrá saber los inodes que quedan libres

> df -hi /

Filesystem	Inodes	IUsed	IFree	IUse%	Mounted on
LABEL=ROOT	64K	15K	50K	23%	/

3. ¿Cuál es el tamaño del directorio /etc? ¿Y el del directorio /var? Compara estos tamaños con los de los directorios /bin, /usr y /lib. Anota brevemente tus conclusiones.

> du /etc --> ficheros de configuración, scripts de arranque, etc.

21064 /etc

> du /var --> archivos variables como archivos de registros y bases de datos

13784 /var

> du /bin --> aplicaciones binarias importantes

5384 /bin

> du /usr --> aplicaciones y archivos a los que puede acceder la mayoría de los usuarios

303176 /usr

> du /lib --> librerías del sistema (libraries)

24540 /lib

(Ejemplo usando du combinado con -sh)

du -sh /lib

24M /lib

**4. Obtén el número de bloques de tamaño 4 KB que utiliza la rama de la estructura jerárquica de directorios que comienza en el directorio /etc. En otras palabras, los bloques de tamaño 4 KB del subárbol cuya raíz es /etc.**

**¿Cuál es el tamaño de bloque, por omisión, utilizado en el SA?**

> df -h /etc

Filesystem	Size	Used	Avail	Use%	Mounted on
LABEL=ROOT	1008M	451M	507M	48%	/

**Ejercicio: Construye los mismos enlaces, duros y simbólicos, que muestra la salida por pantalla anterior.**

**Para ello crea los archivos archivo.txt y target\_hardLink2.txt y, utilizando el manual en línea para ln, construye los enlaces softLink, hardLink y hardLink2. Anota las órdenes que has utilizado.**

> touch archivo.txt --> Crea un archivo vacío

> touch target\_hardLink2.txt

> ln -s archivo.txt softLink

> ln -P archivo.txt hardLink

> ln -P target\_hardLink2.txt hardLink2

**¿Por qué el contador de enlaces del archivo archivo.txt vale 2 si sobre él existen un enlace duro hardLink y un enlace simbólico softLink?**

El contador de enlaces solo mantiene aquellos que sean duros, ya que los simbólicos no comparten el inodo con el archivo original. No obstante, solo por crear un archivo ya se le asocia un enlace duro.

Si además se le crea otro, el archivo contendrá dos.

**Ejercicio: Proporciona las opciones necesarias de la orden ls para obtener la información de metadatos de los archivos de un directorio concreto en los dos casos siguientes:**

- En el caso de que haya archivos de tipo enlace simbólico, la orden debe mostrar la información del archivo al que enlaza cada enlace simbólico y no la del propio archivo de tipo enlace simbólico. → `ls -lL`
- En el caso de enlaces simbólicos debe mostrar la información del enlace en sí, no del archivo al cual enlaza. En el caso de directorios no debe mostrar su contenido sino los metadatos del directorio. → `ls -l`

**Ejercicio:** Consulta el manual en línea para la orden `mknod` y crea un dispositivo de bloques y otro de caracteres. Anota las órdenes que has utilizado y la salida que proporciona un `ls -li` de los dos archivos de dispositivo recién creados. Puedes utilizar las salidas por pantalla mostradas en esta sección del guión para ver el aspecto que debe presentar la información de un archivo de dispositivo.

La orden **`mkmod`** permite especificar el nombre del archivo y los números principal (major) y secundario (minor). Estos números permiten identificar a los dispositivos en el kernel, concretamente en la Tabla de Dispositivos. El número principal determina el controlador al que está conectado el dispositivo y el número secundario al dispositivo en sí.

```
> mknod bloques b 4 30
```

```
> mknod caracteres c 5 40
```

→ `ls -li -->` la b (bloques) y c (caracteres) después del primer número indican qué tipo de archivos especiales son:

```
total 0
6036 -rw-r--r-- 2 root root 0 Oct 11 10:54 archivo.txt
6091 brw-r--r-- 1 root root 4, 30 Oct 11 11:01 bloques
6096 crw-r--r-- 1 root root 5, 40 Oct 11 11:01 caracteres
...
```

## Repaso Sesión 3:

1. Indique la orden necesaria para listar los archivos de nuestro directorio home ordenados según su último acceso → `ls -ltu $HOME`
2. ¿Cómo podríamos ver únicamente el número de usuarios que hay trabajando en el sistema? → `uptime | cut -d "," -f2`
3. ¿Cómo podemos ver únicamente el tiempo que lleva en marcha el sistema? → `uptime -p`
4. Indique la orden para visualizar los bloques libres del SA de un dispositivo cualquiera (ejemplo: `/dev/sda1`) → `tune2fs -l /dev/sda1 | egrep -l "free blocks"`
5. ¿Cómo podríamos saber el tamaño de nuestro directorio home en formato entendible? → `du -h /home` (si haces `du -h | tail -1` solo sale la información de `/home`)



# Sesión 4. Automatización de tareas

## Procesos demonio

Un "*Demonio*" (*Daemon*) en Linux es un script. Un proceso que normalmente está cargado en memoria esperando una señal para ser ejecutado. No quiere esto decir que ocupen CPU.

Normalmente cada daemon tiene un shell script situado en la carpeta */etc/init.d* que permiten iniciarlo, pararlo o ver su estado.

Para lanzar un daemon hay que usar el **comando start**, y por tanto, para pararlo, se usará **stop**. El comando restar hace que se reinicie el daemon, haciendo así que vuelva a leer los archivos de configuración.

Si se quiere ejecutar un daemon habrá que llamarlo desde su ruta, y pasarle los parametros que interesen.

Fundamentalmente, los "Demonio" son programas que ofrecen servicios al resto del sistema. Generalmente se inician durante el arranque, siendo las funciones más comunes de estos las de ofrecer servicios a otros programas, ya sea respondiendo a las peticiones que llegan a través de la red o atendiendo a procesos que se ejecutan en el mismo sistema, así como responder ante cierta actividad del hardware.

## Características

- Ejecución en background (segundo plano) sin estar asociado a un terminal o proceso login
- La mayoría se inician al arrancar el sistema y permanecen mientras este esté encendido. Otros solo se ejecutaran cuando sea necesario
- Si finalizan por algún imprevisto, suele existir un mecanismo que lo detecte y lo re arranque.
- En muchos casos está a espera de un evento (que el daemon sea un servidor, esperando una petición para realizar un servicio)
- Puede ser que su labor sea realizar una tarea periódica, o cuando se cumpla cierta condición
- No suele hacer el trabajo directo, si no que lanza otros procesos para dicho cometido
- Son programas y no parte del kernel (filosofía de la Modularidad de Unix)
- En muchos casos se ejecuta con privilegio superuser(UID=0) cuyo padre es el proceso init (PID=1)

## Ejecutar tareas a una determinada hora: DEMONIO ATD

Con el demonio atd se puede ejecutar una orden en un tiempo especificado. Para interactuar con el demonio se usará

- **at** → Ordenar la ejecución de órdenes a una determinada hora
- **atq** → Consultar la lista de ordenes

- **atrm** → Eliminar ordenes
- **batch** → Ordenar la ejecución de órdenes que se ejecutarán cuando la carga del sistema sea baja

## **Orden AT**

***at [-q queue] [-f <script>] [-mldbv] TIME***

Lee órdenes de la entrada standar o del archivo <script> y provoca su ejecución en la fecha u hora especificada.

Ejemplos:

*Genera una lista de archivos del directorio home en un archivo listahome a las 17:10*

*at 17:10*

*at> ls ~> listahome*

-----

*Programar una tarea a las 10 de la noche*

*> at 10:00 PM*

*warning: commands will be executed using /bin/sh*

*at> sh copia-seguridad.sh*

*at>*

*job 1 at Mon Jul 1 22:00:00 2019*

-----

*listar ordenes*

*> atq*

*1 Mon Jul 1 22:00:00 2019 a calculadora*

*2 Tue Jul 2 07:00:00 2019 a minecraft*

*> atrm 2*

*> atq*

*1 Mon Jul 1 22:00:00 2019 a calculadora*

-----

*Programar una tarea para una fecha*

*> at 11:00 AM April 14*

*> at 10:00 AM 6/22/2021*

*" " tomorrow*

*> at now + 30 minutes*

*> at 7:15am Sep 11*

En resumen, las formas de indicar una hora.

***at -f [SCRIPT] [FORMATOS-TIEMPO]***

**[FORMATOS-TIEMPO]** sustituir por

- *Hora:*

9pm

20:45

now → ahora

noon → 12pm

midnight → 00:00

- *Fecha:*

today → hoy

tomorrow → mañana

Sep 10 → 10 de septiembre

Jan 5 → 5 de enero

También pueden hacerse incrementos:

+ **[NUMERO] minutes** --> Minutos

+ **[NUMERO] hours** --> Horas

+ **[NUMERO] days** --> Dias

+ **[NUMERO] weeks** --> Semanas

+ **[NUMERO] months** --> Mes

+ **[NUMERO] years** --> Año

## **Sobre el entorno de ejecución de las órdenes**

Cuando llegue el momento especificado, se lanzará una shell */bin/sh* para ejecutar el script o el conjunto de ordenes tomadas en la entrada.

## **Salida estándar y salida de error estándar**

Si se lanza la ejecución de un script se tendrá como entrada estándar el teclado; y la pantalla para la salida estándar y salida de error del terminal que se esté usando.

Sin embargo, al lanzar la ejecución asíncrona de una tarea con la orden *AT*, NO se está creando un proceso hijo de la shell.

El nuevo proceso NO tendrá como entrada estándar, salida estándar y de error los asociados a la terminal.

Cuando la orden lanzada de forma retardada sea ejecutada, lo que se genere en la salida estándar y la salida de error estándar se envía al usuario que envió la orden como un correo electrónico usando la orden */usr/bin/sendmail* (la ubicación concreta puede depender de la instalación), si el servicio está disponible.

**\*Nota:** Tener cuidado con que se genere un número excesivo de salidas que creen problemas de espacio.

***at> tar cvf /backups/backup.tar . 1>> ~/salidas 2> /dev/null***

Se redirigen las salidas de error a "null" para que se pierdan y no inunden el correo o a un archivo para poderse consultar en caso de necesidad.

### ***Errores frecuentes:***

Se ha lanzado una orden sin redirigir la salida de error. Pasará totalmente desapercibido el error, y no habrá resultado alguno de la ejecución.

Crear un script sin darle permisos de ejecución, y no tener a la vista un "permiso denegado" que avise que no es posible la ejecución.

No tener incluida la lista de búsqueda al directorio actual. Invocar un script que esté en el directorio actual sin añadir ./ dará error.

Si sendmail está funcionando, se recibirá un correo con el error. Pero, de no ser el caso, deberá de hacerse una redirección manual.

## **Orden BATCH**

Equivale a AT salvo porque no se especifica la hora de ejecución. El trabajo especificado se ejecutará cuando la carga de trabajos del sistema esté por debajo de cierto valor umbral indicado al lanzar atd.

## **Orden AT: Trabajando con colas**

Gestiona distintas colas de trabajos que esperan ser ejecutados. Se designan con una única letra (a-z, A-Z). La a para cola por omisión, b para los trabajos batch. De c en adelante son colas con menos prioridad.

La forma de enviar un trabajo a una cola se hace con el comando at, especificando la cola tras usar -q.

Para visualizar los trabajos de una cola se usa la orden --> ***atq -q c***

## **Aspectos de administración del demonio ATD**

La orden AT podrá ser usada por aquellos usuarios determinados en los archivos siguientes

+ /etc/at.deny → Si un usuario está en dicha lista, no podrá usar el comando

+ /etc/at.allow → Contiene el nombre de los usuarios habilitados (uno por línea)

Puede darse el caso de que no exista ningún archivo. Los usuarios que podrán usar el comando estará determinado por el Sistema y su configuración.

## **Ejecuciones periódicas: DEMONIO CRON**

Este demonio es responsable de ejecutar órdenes con una periodicidad determinada. Para ello será necesario crear un archivo "crontab" con un formato específico (formato crontab).

La orden **crontab** es el intermediario para comunicar dicho archivo "crontab" con el demonio cron.

## **Formato de los archivos CRONTAB: Especificando órdenes**

Cada línea de código de un archivo crontab (excepto los comentarios que están precedidos por #) puede contener estos campos (que representan una orden):

**minuto hora día-del-mes mes día-de-la-semana orden**

MINUTO --> [0-59]

HORA --> [0-23]

DIA --> [0-31]

MES --> [0-12]

DIA SEMANA --> [0-6]. 0 es domingo

SCRIPT --> /DIRECTORIO/script

Cada campo puede contener la siguiente información:

- Un ASTERISCO, \* → Cualquier valor posible
- Un número entero → Activa ese valor determinado
- Dos enteros separados por un guion → Indican un rango de valores
- Una serie de enteros o rangos separados por una coma → Activando cualquier valor de los

que aparecen en la lista.

Ejemplo:

*1 20 \* \* 1-5 [ORDEN/SCRIPT] --> Se debe indicar la ruta, por ejemplo /home/script.sh*

*esto significa "a las 20 horas y 1 minuto de lunes a viernes"*

**\*Nota:** Es necesario conceder permisos de ejecución al script (*chmod a+x [SCRIPT]*), si no puede dar error cuando CRONTAB intente ejecutarlo)

## **Orden CRONTAB**

Se encarga de instalar, desinstalar o listar los trabajos que procesará el demonio cron. La sintaxis para especificar el archivo <file> como archivo con formato "crontab" que contiene la lista de trabajos para cron es:

***crontab [options] [ARCHIVO]***

***Ejemplo: crontab -e jones → Esto crea un fichero crontab para el usuario jones.***

Para saber si un fichero crontab ya existe se ejecuta:

***ls -l /var/spool/cron/crontabs (Ejemplo de Internet)***

```
-rw-r--r-- 1 root sys 190 Feb 26 16:23 adm
-rw----- 1 root staff 225 Mar 1 9:19 jones
-rw-r--r-- 1 root root 1063 Feb 26 16:23 lp
...
```

Cada usuario puede contar con su propio archivo crontab donde añadir los comandos o programas que tendrán un periodicidad asignada para su ejecución con un formato igual al indicado en el apartado anterior.

## **Información relativa a Crontab:**

Para remplazar el archivo existente por otro que defina el usuario se debe utilizar el siguiente comando:

***crontab [ARCHIVO]*** → Crea un archivo crontab para el usuario que invoque la orden-

- ***crontab -e*** → Para editar el archivo existente en la actualidad.

Si se usa ***crontab -e [USER]***, se manejará el archivo del usuario en cuestión

- ***crontab -l*** → Listar todas las tareas existentes en el crontab del usuario.

- ***crontab -d*** → Borrar el crontab que está configurado.

- ***crontab -c [DIRECTORIO]*** → Definir el directorio en el que se almacenará el archivo de crontab. Para realizar esta operación se deben tener permisos de ejecución en dicho directorio.

- ***crontab -u [USER]*** → Maneja el crontab de otros usuarios existentes en el sistema. (Hay que tener permisos de administrador).

Otras formas de usarlo:

> ***crontab -l -u [USER]***

> ***crontab -e [USER]***

> ***crontab -d -u [USER]***

En el archivo crontab incluso pueden ponerse órdenes de la forma:

→ ***rm /home/archivo.txt***

→ `python /home/script.py`

Y ejecutará comando o programas, o procesos de compilación. Cualquier orden que el intérprete sepa procesar.

### **Formato de los archivos crontab: Especificando variables de entorno**

Una línea en un archivo crontab puede contener dos cosas:

- Una línea de especificación de una orden (con el formato dicho anteriormente)
- Una línea de asignación de valores a variables de entorno; **[NOMBRE]=[VALOR]**

Algunas ya están definidas por defecto:

- *SHELL* → `/bin/bash`
- *LOGNAME* y *HOME* → Tomadas del archivo `/etc/passwd`

A la hora de asignar valores no se realiza sustitución.

### **Aspectos de administración del demonio crontab**

Al final que ocurre con el demonio atd, este también cuenta con dos archivos que establecen qué usuarios están capacitados para su uso:

`/etc/cron.deny` → Aquellos usuarios listados (uno por línea) no tendrán permiso para ejecutarlo.

`/etc/cron.allow` → Los usuarios que aparezcan en este archivo estarán capacitados para ejecutar dicho comando.

### ***NOTA IMPORTANTE DE LOS DEMONIOS:***

Los demonios no tienen terminal asociada, por eso, tenemos que indicar la salida de lo que hemos programado:

`at now + 1 minuto`

`at> ls 1>> salida_ls.txt`

Si se usa '`>`' sobrescribe y si se usa '`>>`' concatena.

Si es '`1>`' se trata de la salida estándar, y '`2>`' es para cuando exista una salida errónea.

## Actividades Sesión 4:

**Ejercicio:** A partir de la información proporcionada por la orden `ps` encuentre los datos asociados a los demonios `atd` y `crond`, en concreto: quién es su padre, qué terminal tienen asociado y cuál es su usuario.

**Nota:** Si el demonio `atd` no está instalado en su sistema puede descargarlo del repositorio: <http://rpm.pbone.net> con nombre

*at-3.1.12-5.fc14.i686.rpm. Hay que instalarlo y arrancar el servicio con la orden “service atd start”*

`ps -l`

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
4	S	0	1189	1188	0	80	0	-	820	wait	tty0	00:00:00	bash
4	R	0	1240	1189	0	80	0	-	655	-	tty0	00:00:00	ps

Como no ha podido instalarse en Fedora, y mi equipo es de 64 bits, tampoco lo he podido instalar en mi propio sistema pero debería salir una información que confirme que: El usuario es `daemon`, y el padre es 1, sin tener terminal alguno asociado '?'

Con el demonio `cron` ocurre lo mismo, no ha podido ejecutarse pero la información que debería mostrar es el que padre es 1, el usuario que lo invoca es `root`, y no tiene terminal asociada, '?'

**Ejercicio:** Crea un archivo `genera-apunte` que escriba la lista de hijos del directorio `home` en un archivo de nombre `listahome-`date +%Y-%j-%T-$``, es decir, la yuxtaposición del literal “listahome” y el año, día dentro del año, la hora actual y pid (consulte la ayuda de `date`).

```
#!/bin/sh
fecha=`date +%Y-%j-%T-$`
ls >> listahome-$fecha
```

Para poder ejecutarlo, hay que recordar que es necesario darle permiso de ejecución

```
> chmod a+x genera-apunte.sh
```

Y para ejecutarlo

```
> ./genera-apunte.sh
```

Lanza la ejecución del archivo `genera-apunte` un minuto más tarde de la hora actual.

```
at -f ./genera-apunte.sh now + 1 minutes
```

**¿En qué directorio se crea el archivo de salida?** → Se crearía en el directorio donde se lanzó el demonio.

**Ejercicio:** Lanza varias órdenes `at` utilizando distintas formas de especificar el tiempo como las siguientes (será de utilidad la opción `-v`):

**a) a medianoche de hoy**

```
> at -f [SCRIPT] midnight
```



**b) un minuto después de la medianoche de hoy**

> at -f [SCRIPT] midnight + 1 minutes

**c) a las 17 horas y 30 minutos de mañana**

> at -f [SCRIPT] tomorrow 17:30

**d) a la misma hora en que estemos ahora pero del día 25 de diciembre del presente año**

> at -f [SCRIPT] 12/25/2020

**e) a las 00:00 del 1 de enero del presente año**

No se pueden programar tareas para el pasado.

**Ejercicio: El proceso nuevo que se lanza al cumplirse el tiempo que se especificó en la orden at....**

**a) ¿Qué directorio de trabajo tiene inicialmente? ¿hereda el que tenía el proceso que invocó a at o bien es el home, directorio inicial por omisión?**

Hereda el directorio que tenía el proceso que invocó a at

**b) ¿Qué máscara de creación de archivos umask tiene? ¿es la heredada del padre o la que se usa por omisión?**

La máscara heredada

**\*Nota: umask o User Mask**, la máscara de usuario usada para establecer los permisos a un fichero o directorio recién creado. Al usar umask, devuelve el valor de umask configurado en el sistema. Normalmente su valor suele ser 022. Para cambiarlo basta con ejecutar umask [VALOR]. Si se cambia dicha máscara, al crear un archivo lo hará con esos nuevos permisos.

EL CAMBIO DE UMASK SOLO PERMANECE DURANTE LA SESIÓN DE LA TERMINAL. SI SE CIERRA Y LUEGO SE VUELVE A ABRIR EL VALOR VOLVERÁ A SER EL DE POR DEFECTO.

**c) ¿Hereda las variables locales del proceso padre?**

Las únicas que no hereda son BASH\_VERSION, DISPLAY, EUID, GROUPS, SHELL\_OPTS, TERM, UID.

**Ejercicio: El proceso nuevo que se lanza al cumplirse el tiempo que se especificó en la orden at.... ¿de quién es hijo?**

```
#!/bin/bash
nombrearchivo=`date +%Y-%j-%T`
ps -ef > $nombrearchivo
echo Mi pid = $$ >> $nombrearchivo
```

```
> vi procesos-existentes
> chmod a+x procesos-existentes
> ./procesos-existentes
> cat 2019-286-06\30\00
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	05:06	?	00:00:00	/sbin/init
root	2	0	0	05:06	?	00:00:00	[kthreadd]
root	3	2	0	05:06	?	00:00:01	[ksoftirqd/0]
root	4	2	0	05:06	?	00:00:00	[kworker/0:0]
root	5	2	0	05:06	?	00:00:00	[kworker/u:0]
...							
root	1332	1189	0	06:30	tty0	00:00:00	/bin/bash ./procesos-existentes
root	1334	1332	0	06:30	tty0	00:00:00	ps -ef

Mi PID = 1332

**Ejercicio:** Construye un script que utilice la orden **find** para generar en la salida estándar los archivos modificados en las últimas 24 horas (partiendo del directorio **home** y recorriéndolo en profundidad), la salida deberá escribirse el archivo de nombre “modificados\_<año><día><hora>” (dentro del directorio **home**). Con la orden **at** provoque que se ejecute dentro de un día a partir de este momento.

**\*Nota:** uso de **FIND**. Para encontrar ficheros por su edad, podríamos hacerlo también con tres parámetros específicos para estas tareas. Por **tiempo de acceso (-atime)**, por **última modificación (mtime)** o por el **último cambio de fichero (-ctime)**. Todas las cifras, expresadas en días, cuidado.

Por ejemplo, busquemos ficheros que no hayan sido modificados en los dos últimos días

***find -mtime +2***

Se pueden combinar las opciones y buscar ficheros en un rango, por ejemplo, ficheros que tengan entre 2 y 5 días de edad:

***find -mtime +2 -mtime -5***

La opción **-mtime** filtra los resultados para que sólo se obtengan los archivos/directorio cuyos datos fueron modificados por última vez hace **n** días (**n\*24** horas). Los campos numéricos son especificados de la forma:

***+ [N] --> busca valores Mayor que [N]***

***- [N] --> busca valor Menor que [N]***

***[N] --> busca valores Exactamente [N]***

Buscar todos los archivos que hayan cambiado en los últimos 30 minutos:

***find / -mmin -30 -type f***

los modificados exactamente hace 30 minutos: (**/ --> busca desde la raíz**)

***find / -mmin 30 -type f***

***find . -mtime 0 (busca archivos modificados entre ahora y hace un dia)***

***find . -mtime -1 (busca archivos modificados hace menos de un dia)***

***find . -atime 1 (busca archivos accedidos entre hace 24 y 48 horas)***

*find . -ctime +1 (busca archivos cuyo status haya cambiado hace más de 48 horas)*

Tras el estudio de esto, la solución que se puede dar al ejercicio es:

```
#!/bin/bash
fecha=$(date +"%Y-%d-%T")
fichero="modificados-$fecha"
#-mtime indica fecha de modificacion en las ultimas n*24
#se pone MENOS uno para multiplicar por 1*24 e indicar
#hace "valores menor a 1*24"
find /home -mtime -1 >> $fichero
```

at -f ./buscar-modificados.sh tomorrow

**Ejercicio: Construye tres script que deberás lanzar a las colas c, d y e especificando una hora concreta que esté unos pocos minutos más adelante (no muchos para ser operativos). Idea qué actuación deben tener dichos script de forma que se ponga de manifiesto que de esas colas la más prioritaria es la c y la menos es la e. Visualiza en algún momento los trabajos asignados a las distintas colas.**

Hasta ahora se han creado tres scripts

buscar-modificados.sh; genera-apunte.sh; procesos-existentes.sh

Para añadir los procesos a una cola se usará la orden y opciones siguientes:

```
at -q e -f ./buscar-modificados.h now + 1 minutes
at -q d -f ./genera-apunte.sh now +1 minutes
at -q c -f ./procesos.existentes.sh now +1 minutes
at -q c --> Visualizará los trabajos de la cola c
```

**Ejercicios: Construye un script que sea lanzado con una periodicidad de un minuto y que borre los nombres de los archivos que cuelguen del directorio /tmp/varios y que comiencen por "core" (cree ese directorio y algunos archivos para poder realizar esta actividad).**

**Utiliza la opción -v de la orden rm para generar como salida una frase de confirmación de los archivos borrados; queremos que el conjunto de estas salidas se añadan al archivo /tmp/listacores.**

**Prueba la orden crontab -l para ver la lista actual de trabajos (consulte la ayuda para ver las restantes posibilidades de esta orden para gestionar la lista actual de trabajos).**

Para indicar que se quieren borrar los archivos que empiecen por core se usa rm de la forma:

```
rm /DIRECTORIO/core*
mkdir /tmp/varios --> Se crea el directorio
rm -v /tmp/varios/core* >> /tmp/listacores --> de esta forma se añaden las salidas al archivo
```

Script -> borra-cores.sh:

```
#!/bin/bash
rm -v /tmp/varios/core* >> /tmp/listacores
```

> chmod a+x borra-cores.sh

Y en el archivo crontab añadir la siguiente linea

> crontab -e

\* \* \* \* \* /home/borra-cores.sh

# Examen 2019-2020

**Ejercicio: [Usuario root].** Una forma de averiguar si el directorio cuya ruta está almacenada en \$d está siendo utilizada como punto de montaje es ejecutar `ls -ld $d` y chequear el primer carácter. ¿Qué indica el tipo?

mkdir prueba (en /home)

`ls -ld prueba`

`drwxr-xr-x 2 root root 4096 Nov 10 14:13 prueba`

—> Tipo de fichero:

- directorio

d

- archivo:

-

- enlace simbólico:

l

- socket

s

- tubería

p

- archivo de bloque

b

- archivo de caracteres

c

El primer carácter es una 'd', la cual es de tipo directorio; por lo que puede servir de punto de montaje.

**Ejercicio: [Ubuntu]** Ejecuta “top” para mostrar los procesos creados por el usuario actual.

```
alua
alba@albauwu:~$ top | grep -i alba
2423 alba 20 0 24,5g 345428 117884 S 93,8 2,8 20:55.08 Discord
1572 alba 9 -11 2876816 22528 17312 S 6,2 0,2 1:50.96 pulseaudio
1654 alba 20 0 924856 98932 66824 S 6,2 0,8 1:15.07 Xorg
2245 alba 20 0 831840 130544 86916 S 6,2 1,1 0:21.05 Discord
3872 alba 20 0 20720 4084 3272 R 6,2 0,0 0:00.01 top
```

top -u alba o top -u \$USER ya que \$USER indica el usuario actual.

```
alba@albauwu:~$ top -u alba
top - 20:27:59 up 22 min, 1 user, load average: 2,81, 2,33, 1,81
Tareas: 251 total, 1 ejecutar, 240 hibernar, 10 detener, 0 zombie
%Cpu(s): 31,1 usuario, 3,6 sist, 0,0 adecuado, 64,0 inact, 0,4 en espera, 0,0 hardw int, 0,8 softw i
MiB Mem : 11899,2 total, 8743,6 libre, 1331,3 usado, 1824,4 búfer/caché
MiB Intercambio: 2048,0 total, 2048,0 libre, 0,0 usado. 10011,2 dispon Mem

  PID USUARIO PR NI VIRT RES SHR S %CPU %MEM HORA+ ORDEN
2423 alba 20 0 24,5g 346364 117884 S 123,2 2,8 24:18.12 Discord
1572 alba 9 -11 3139288 23020 17584 S 8,6 0,2 2:09.02 pulseaudio
1654 alba 20 0 928408 100016 67272 S 6,0 0,8 1:33.41 Xorg
2851 alba 20 0 900648 55040 41076 S 2,0 0,5 0:06.52 gnome-terminal-
1891 alba 20 0 4403960 270756 107236 S 1,7 2,2 1:44.09 gnome-shell
2245 alba 20 0 831840 130696 86916 S 0,7 1,1 0:23.07 Discord
4191 alba 20 0 20720 3908 3304 R 0,3 0,0 0:00.09 top
1552 alba 20 0 10234 10744 8260 S 0,0 0,1 0:00.53 customd
```

**Ejercicio: [Ubuntu] Suponemos que no existen en nuestro sistema grupos con el mismo nombre que los usuarios existentes, hemos ejecutado:**

**grep -w \$USER /etc/group | cut -d: -f1**

**Elige:**

- A) Nombre del grupo principal del usuario que ejecuta el shell.**
- B) Nombre de los usuarios que pertenecen al grupo principal del usuario que ejecuta el shell.**
- C) Nombre de todos los grupos a los que pertenece el usuario que ejecuta el shell**
- D) Ninguna**

```
1000 alba 20 0 248088 7030 0984 S 0,0 0,1 0,0
alba@albauwu:~$ grep -w $USER /etc/group | cut -d: -f1
adm
cdrom
sudo
dip
plugdev
lpadmin
lxd
alba
sambashare
```

Si ejecutamos la orden sin el cut , podemos ver la información de /etc/group en los que aparezca el valor \$USER.

/etc/group almacena esta información:

## Understanding the /etc/group File

It stores group information or defines the user groups i.e. it defines the groups to which users belong. There is one entry per line, and each line has the following format (all fields are separated by a colon (:))

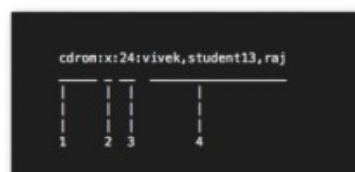


Fig.01: Sample entry in /etc/group file

Where,

- 1. group\_name:** It is the name of group. If you run ls -l command, you will see this name printed in the group field.
- 2. Password:** Generally password is not used, hence it is empty/blank. It can store encrypted password. This is useful to implement privileged groups.
- 3. Group ID (GID):** Each user must be assigned a group ID. You can see this number in your /etc/passwd file.
- 4. Group List:** It is a list of user names of users who are members of the group. The user names, must be separated by commas.

Como hemos cogido el primer campo, estamos haciendo referencia al nombre del grupo. Por lo cual la respuesta correcta es la C.

**Ejercicio: Se ejecuta `ln -s $n1 $n2`. ¿Verdadero o falso?**

- 1) Para que haya tenido éxito `$n1` y `$n2` no deben contener el carácter `/`. → Falso.
- 2) El contador de enlaces duros de `$n1` no ha cambiado. → Verdadero. Porque al usar el `ln -s` estamos creando un enlace simbólico que no se cuenta en el contador de enlaces duro.
- 3) El número de nodo de `$n1` es igual al de `$n2`. → Falso. Porque los enlaces simbólicos no comparten el inodo con el archivo original.
- 4) `$n1` podría ser de tipo directorio o enlace simbólico. → Verdadero.

```
alba@albauwu:~$ cd Documentos/
alba@albauwu:~/Documentos$ ls
dir dirsimb dirsimb2 p1 p1simb p1simb2
alba@albauwu:~/Documentos$ ls -l
total 4
drwxrwxr-x 2 alba alba 4096 nov 10 21:02 dir
lrwxrwxrwx 1 alba alba 3 nov 10 21:03 dirsimb -> dir
lrwxrwxrwx 1 alba alba 14 nov 10 21:03 dirsimb2 -> Documentos/dir
-rw-rw-r-- 1 alba alba 0 nov 10 20:57 p1
lrwxrwxrwx 1 alba alba 2 nov 10 21:02 p1simb -> p1
lrwxrwxrwx 1 alba alba 6 nov 10 21:02 p1simb2 -> p1simb
```

**Ejercicio: Verdadero o falso: “Estando ejecutando el shell desde un usuario sin permisos de superusuario, la orden `$nice -0 ejecutable1` lanza la ejecución de `ejecutable1` con el mayor peso o preferencia posible, respecto a la asignación de la CPU”**

Falso. Al no tener privilegios de root, el usuario solo puede desfavorecer la posibilidad de ejecución de un proceso. Al usar `nice -0` se mantiene la prioridad actual del proceso, por lo que no se le estaría dando mayor peso.

**Ejercicio: Explicar el significado de:**

**`df: /dev/loop0 19000 1100 7% /otros/almacenamiento`**

`/dev/loop0` → sistema de ficheros → El nombre del archivo del sistema.

`19000` → bloques de 1K → Indica el tamaño total de la unidad de almacenamiento.

`1100` → Usados → Muestra cuando un espacio está siendo ocupado en cada archivo del sistema.

`7%` → Uso (%) → Muestra el porcentaje del espacio que está siendo usado.

`/otros/almacenamiento` → Punto de montaje → muestra el directorio donde está cada archivo del sistema.

**Ejercicio: Indique las órdenes para conocer cuál es el PID de los procesos hermanos del proceso cuyo PID hemos almacenado en `$p`.**

Imaginando que `$p = 3982`, primero buscamos cual sería el PPID del padre con la orden `ps -o ppid = -p $p` y después con `ps tree -p -s PPID` buscamos los procesos hermanos.

```
alba@albauwu:~$ ps -o ppid= -p 3982
3844
alba@albauwu:~$ ps tree -p -s 3844
systemd(1)---systemd(1553)---gnome-terminal-(2851)---bash(3844)
└── grep(3982)
    ├── pstree(5899)
    └── top(3981)
```

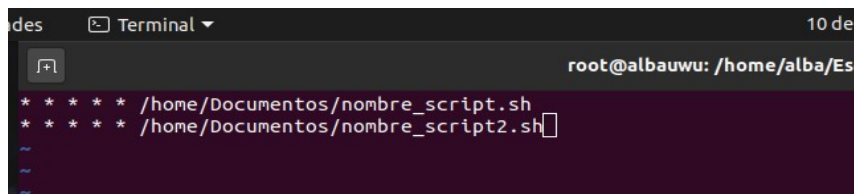


**Ejercicio: [Usuario root]** Estando hecho todo lo previo al montaje, di las órdenes para conseguir que el Sistema de Archivos ubicado en el dispositivo /dev/loop0 esté montado en el directorio /nuevo para lectura y escritura, durante todo el año pero solo de lunes a viernes de 8am a 8pm.

TERMINAL:

\$crontab -e:

En los \* \* \* \* \* se añaden las opciones de cuando se tienen que ejecutar

A terminal window titled 'Terminal' with a dark background. The prompt is 'root@albauwu: /home/alba/Es'. The user has entered two lines of crontab entries: '\* \* \* \* \* /home/Documentos/nombre\_script.sh' and '\* \* \* \* \* /home/Documentos/nombre\_script2.sh'.

```
root@albauwu: /home/alba/Es
* * * * * /home/Documentos/nombre_script.sh
* * * * * /home/Documentos/nombre_script2.sh
```

SCRIPT1:

#!/bin/bash

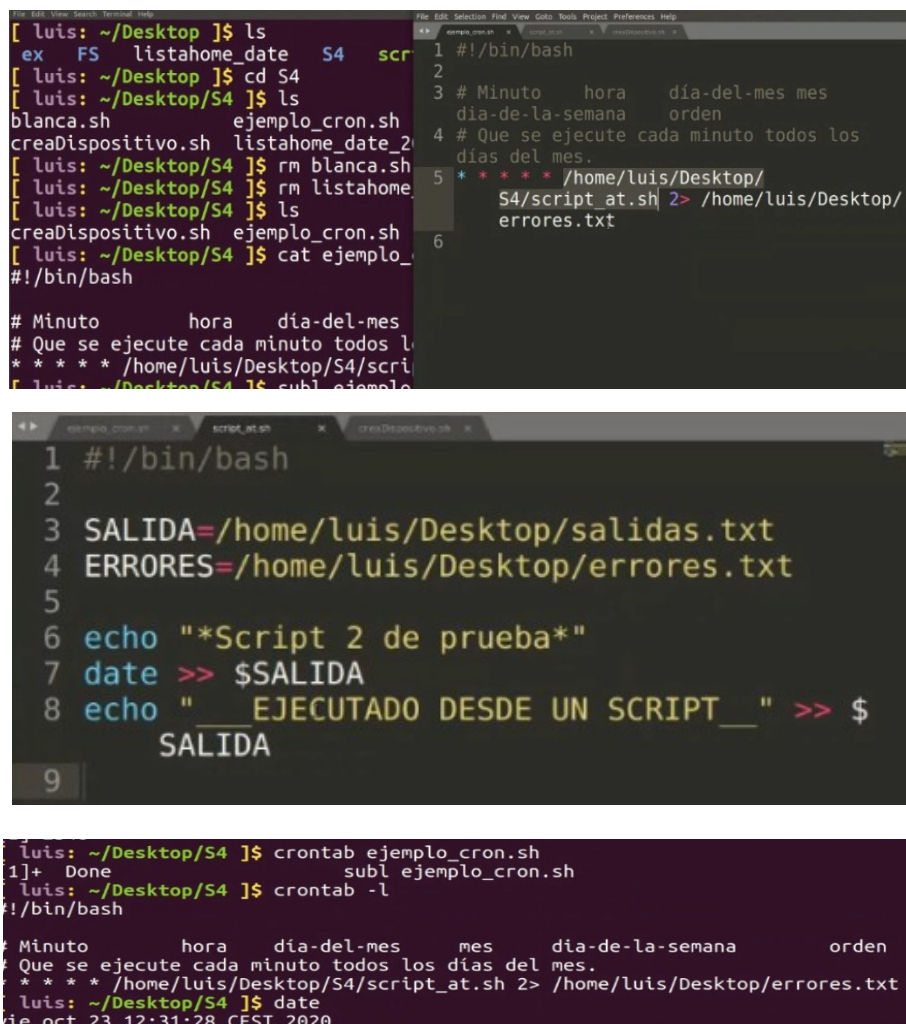
mount /dev/loop0 /nuevo -rw

SCRIPT2:

#!/bin/bash

umount /dev/loop0

### OTRO EJEMPLO CRONTAB:

Three terminal screenshots showing the setup of a cron job. The first screenshot shows file management on the Desktop and S4 directory. The second shows the content of 'ejemplo\_cron.sh' and 'script\_at.sh'. The third shows the crontab being edited and the current date.

```
[ luis: ~/Desktop ]$ ls
ex FS listahome_date S4 scr
[ luis: ~/Desktop ]$ cd S4
[ luis: ~/Desktop/S4 ]$ ls
blanca.sh ejemplo_cron.sh
creaDispositivo.sh listahome_date_2
[ luis: ~/Desktop/S4 ]$ rm blanca.sh
[ luis: ~/Desktop/S4 ]$ rm listahome
[ luis: ~/Desktop/S4 ]$ ls
creaDispositivo.sh ejemplo_cron.sh
[ luis: ~/Desktop/S4 ]$ cat ejemplo_
#!/bin/bash

# Minuto      hora    día-del-mes
# Que se ejecute cada minuto todos los días del mes.
* * * * * /home/luis/Desktop/S4/script_at.sh 2> /home/luis/Desktop/errores.txt
[ luis: ~/Desktop/S4 ]$ subl ejemplo_

1 #!/bin/bash
2
3 # Minuto      hora    día-del-mes
4 # Que se ejecute cada minuto todos los días del mes.
5 * * * * * /home/luis/Desktop/S4/script_at.sh 2> /home/luis/Desktop/errores.txt
6

1 #!/bin/bash
2
3 SALIDA=/home/luis/Desktop/salidas.txt
4 ERRORES=/home/luis/Desktop/errores.txt
5
6 echo "*Script 2 de prueba*"
7 date >> $SALIDA
8 echo "EJECUTADO DESDE UN SCRIPT_" >> $SALIDA
9

[ luis: ~/Desktop/S4 ]$ crontab ejemplo_cron.sh
[ luis: ~/Desktop/S4 ]$ crontab -l
#!/bin/bash

# Minuto      hora    día-del-mes      mes    día-de-la-semana    orden
# Que se ejecute cada minuto todos los días del mes.
* * * * * /home/luis/Desktop/S4/script_at.sh 2> /home/luis/Desktop/errores.txt
[ luis: ~/Desktop/S4 ]$ date
vie oct 23 12:31:28 CEST 2020
```