

## 1. Editor de textos:

(a) Nombre al menos 3 editores de texto que puede utilizar desde la línea de comandos.

- Sublime Text.
- Vim.
- Emacs.
- Gedit.
- pico/nano.

(b) ¿En qué se diferencia un editor de texto de los comandos cat, more o less? Enumeré los modos de operación que posee el editor de textos vi.

## Comando Unix CAT

El comando 'cat' imprimirá por pantalla el contenido del fichero sin ningún tipo de paginación ni posibilidad de modificarlo. Básicamente concatena archivos o la salida estándar en la salida estándar. Podemos pasarle parámetros como:

```
-A, --show-all      lo mismo que -vET
-b, --number-nonblank  numera las líneas que no están vacías
-e                  lo mismo que -vE
-E, --show-ends      muestra un $ al final de cada línea
-n, --number          numera todas las líneas
-s, --squeeze-blank   nunca muestra más de una línea vacía,
-t                  equivalente a -vT
-T, --show-tabs      muestra los caracteres de tabulación como ^I
-u                  (sin efecto)
-v, --show-nonprinting  utiliza la notación ^ y M-, salvo para LFD y TAB
--help              muestra esta ayuda y finaliza
--version            informa de la versión y finaliza
```

Mostrar el contenido de un fichero:

```
$ cat fichero
```

Concatenar dos ficheros de texto en uno:

```
$ cat fichero1 fichero2 > fichero3
```

## Comando Unix MORE

Al igual que 'cat', 'more' permite visualizar por pantalla el contenido de un fichero de texto, con la diferencia con el anterior de que 'more' pagina los resultados. Primero mostrará por pantalla todo lo que se pueda visualizar sin

hacer scroll y después, pulsando la tecla espacio avanzará de igual modo por el fichero.

```
$ more fichero
texto de ejemplo
texto de ejemplo
texto de ejemplo
--Más--(23%)
```

También podemos especificarle el número de líneas a mostrar y otros parámetros:

```
uso: more [-df|pcsu] [+númlíneas | +/-patrón] nombre1 nombre2 ...
```

## Comando Unix LESS

El comando 'less' es el más completo de los tres, pues puede hacer todo lo que hace 'more' añadiendo mayor capacidad de navegación por el fichero (avanzar y retroceder) además de que sus comandos están basados en el editor 'vi', del cual se diferencia en que no tiene que leer todo el contenido del fichero antes de ser abierto. Tiene una gran cantidad de opciones y parámetros, como siempre lo recomendable:

```
$ less --help
```

Existen tres modos o estados de vi:

- Modo comando: este es el modo en el que se encuentra el editor cada vez que se inicia. Las teclas ejecutan acciones (comandos) que permiten mover el cursor, ejecutar comandos de edición de texto, salir de vi, guardar cambios, etc.
- Modo inserción o texto: este es el modo que se usa para insertar el texto. Existen varios comandos que se pueden utilizar para ingresar a este modo.
- Modo línea o ex: se escriben comandos en la última línea al final de la pantalla.

**(c) Nombre los comandos más comunes que se le pueden enviar al editor de textos vi**

Comando	Significado
Empezar vi	
vi nombre_de_archivo	Abrir o crear el archivo
vi	Abrir un archivo nuevo para nombrarlo más tarde

Comando	Significado
vi -r nombre_de_archivo	Recuperar un archivo de una caída del sistema
view nombre_de_archivo	Abrir archivo sólo para leer
Comandos del cursor	
h	Moverse un carácter hacia la izquierda
j	Moverse una línea hacia abajo
k	Moverse una línea hacia arriba
l	Moverse un carácter a la derecha
w	Moverse una palabra a la derecha
W	Moverse una palabra a la derecha (pasados los signos de puntuación)
b	Moverse una palabra a la izquierda
B	Moverse una palabra a la izquierda (pasados los signos de puntuación)
e	Moverse al final de la palabra actual
Return	Moverse una línea hacia abajo
Back Space	Moverse un carácter a la izquierda
Space Bar	Moverse un carácter a la derecha
H	Moverse a la parte de arriba de la pantalla
M	Moverse al centro de la pantalla
L	Moverse a la parte inferior de la pantalla
Ctrl-F	Paginar una pantalla hacia adelante
Ctrl-D	Desplazarse media pantalla hacia adelante
Ctrl-B	Paginar una pantalla hacia atrás

Comando	Significado
Ctrl-U	Desplazarse media pantalla hacia atrás
Insertar caracteres y líneas	
a	Insertar caracteres a la derecha del cursor
A	Insertar caracteres al final de la línea
i	Insertar caracteres a la izquierda del cursor
I	Insertar caracteres al principio de línea
o	Insertar una línea por debajo el cursor
O	Insertar una línea por encima del cursor
Cambiar texto	
cw	Cambiar una palabra (o parte de una palabra) a la derecha del cursor
c	Cambiar una línea
C	Cambiar desde el cursor hasta el final de la línea
s	Sustituir cadena por carácter(es) desde el cursor hacia adelante
r	Reemplazar el carácter marcado por cursor por otro carácter
r Return	Partir una línea
J	Unir la línea actual con la línea inferior
xp	Transponer el carácter del cursor con el carácter a la derecha
~	Cambiar el tipo de letra (mayúscula o minúscula)
u	Deshacer el comando anterior
U	Deshacer todos los cambios en la línea actual

Comando	Significado
:u	Deshacer el comando anterior sobre la línea última
Eliminar texto	
x	Eliminar el carácter del cursor
X	Eliminar el carácter a la izquierda del cursor
dw	Eliminar la palabra (o la parte de la palabra a la derecha del cursor)
dd	Eliminar la línea que contiene al cursor
D	Eliminar la parte de la línea a la derecha del cursor
dG	Eliminar hasta el final de línea
d1G	Eliminar desde el principio del archivo hasta el cursor
:5,10 d	Eliminar las líneas de la 5 a la 10
Copiar y mover texto	
yy	Tirar o copiar línea
Y	Tirar o copiar línea
p	Poner la línea tirada o eliminada por debajo de la línea actual
P	Poner la línea tirada o eliminada por encima de la línea actual
:1,2 co 3	Copiar las líneas de la 1 a la 2 y ponerlas después de la línea 3
:4,5 m 6	Mover las líneas de la 4 a la 5 y ponerlas después de la línea 6
Ajustar la numeración de las líneas	
:set nu	Mostrar los números de las líneas

Comando	Significado
:set nonu	Esconder los números de las líneas
	Establecer la distinción entre mayúsculas y minúsculas
:set ic	En la búsqueda se ignora la distinción entre mayúsculas y minúsculas
:set noic	En la búsqueda se distingue entre mayúsculas y minúsculas
Encontrar una línea	
G	Ir a la última línea del archivo
1G	Ir a la primera línea del archivo
21G	Ir a la línea 21
Buscar y reemplazar	
/string	Búsqueda de <b>cadena de caracteres</b>
?string	Búsqueda hacia atrás de <b>cadena de caracteres</b>
n	Encontrar la siguiente aparición de <b>string</b> en la dirección de búsqueda
N	Encontrar la aparición previa de la <b>cadena de caracteres</b> en la dirección de búsqueda
:g/search/s//replace/g	Buscar y reemplazar
Limpiar la pantalla	
Ctrl-L	Limpiar (actualizar) la pantalla
	Insertar un archivo en otro archivo
:r nombre_de_archivo	Insertar (leer) el archivo a continuación del cursor
:34 r nombre_de_archivo	Insertar el archivo después de la línea 34
Guardar y salir	

Comando	Significado
:w	Guardar los cambios (escribir el contenido de la memoria intermedia)
:w nombre_de_archivo	Escribir el contenido de la memoria intermedia a un archivo con nombre
:wq	Guardar los cambios y salir de vi
ZZ	Guardar los cambios y salir de vi
:q!	Salir sin guardar los cambios

## Editor de textos vim

- Presente en cualquier distribución de GNU/Linux
- Posee 3 modos de ejecución:
  - Modo Insert (**Ins** o **i**)
  - Modo Visual (**v**)
  - Modo de Órdenes o Normal (**Esc**)
- Se le puede enviar una serie de comandos útiles
  - **w**: escribir cambios
  - **q** ó **q!**: salir del editor
  - **dd**: cortar
  - **y**: copiar al portapapeles
  - **p**: pegar desde el portapapeles
  - **u**: deshacer
  - **/frase**: busca "frase" dentro del archivo

## 2. Proceso de Arranque SystemV :

(a) Enumere los pasos del proceso de inicio de un sistema GNU/Linux, desde que se prende la PC hasta que se logra obtener el login en el sistema.

- ① Se empieza a ejecutar el código del BIOS
- ② El BIOS ejecuta el POST
- ③ El BIOS lee el sector de arranque (MBR)
- ④ Se carga el gestor de arranque (MBC)
- ⑤ El bootloader carga el *kernel* y el *initrd*
- ⑥ Se monta el *initrd* como sistema de archivos raíz y se inicializan componentes esenciales (ej.: scheduler)
- ⑦ El Kernel ejecuta el proceso *init* y se desmonta el *initrd*
- ⑧ Se lee el `/etc/inittab`
- ⑨ Se ejecutan los scripts apuntados por el *runlevel 1*
- ⑩ El final del *runlevel 1* le indica que vaya al *runlevel* por defecto
- ⑪ Se ejecutan los scripts apuntados por el *runlevel* por defecto
- ⑫ El sistema está listo para usarse

**(b) Proceso INIT. ¿Quién lo ejecuta? ¿Cuál es su objetivo?**

Lo ejecuta el Kernel y su función es cargar todos los subprocesos necesarios para el correcto funcionamiento del SO

Su función es cargar todos los subprocesos necesarios para el correcto funcionamiento del SO. Posee PID 1 y se encuentra en `/sbin/init`. Se lo configura a través del archivo `/etc/inittab`. No tiene padre y es padre de todos los procesos. Es el encargado de montar los filesystems y de hacer disponibles los demás dispositivos.

**(c) Ejecute el comando `ps tree`. ¿Qué es lo que se puede observar a partir de la ejecución de este comando?**

Que muestra los procesos en ejecución en forma de árbol (padre e hijo) siendo `systemd` el padre de todos los procesos.

**(d) RunLevels. ¿Qué son? ¿Cuál es su objetivo?**

Es el modo en que arranca linux (3 en redhat, 2 en Debian)

Un nivel de ejecución es básicamente una configuración de programas y servicios que se ejecutarán orientados a un determinado funcionamiento.

El proceso de arranque se divide en niveles. Cada uno es responsable de levantar o bajar una serie de servicios.

Se encuentran definidos en `/etc/inittab`:

*id:niveles\_ejecucion:acción:proceso*

Id: identifica la entrada en `inittab` (1 a 4 caracteres)

\* Niveles\_ejecucion: el/los nivel de ejecución en los que se realiza la acción



\* Acción: describe la acción a realizar- wait: Se inicia cuando se entra al runlevel e init espera a que termine- initdefault- ctrlaltdel:se ejecutará cuando init reciba la señal SIGINT- off, repawn, once, boot, bootwait, powerwait, otras...

\* Proceso: el proceso exacto que será ejecutado

**(e) ¿A qué hace referencia cada nivel de ejecución según el estándar? ¿Dónde se define qué Runlevel ejecutar al iniciar el sistema operativo? ¿Todas las distribuciones respetan estos estándares?**

- Existen 7, y permiten iniciar un conjunto de procesos al arranque o apagado del sistema
- Según el estándar:
  - **0**: halt (parada)
  - **1**: single user mode (monousuario)
  - **2**: multiuser, without *NFS* (modo multiusuario sin soporte de red)
  - **3**: full multiuser mode console (modo multiusuario completo por consola)
  - **4**: no se utiliza
  - **5**: X11 (modo multiusuario completo con login gráfico basado en X)
  - **6**: reboot

- Los scripts que se ejecutan están en `/etc/init.d`
- En `/etc/rcX.d` (donde  $X = 0..6$ ) hay links a los archivos del `/etc/init.d`

- Formato de los links:  
`[S|K]<orden><nombreScript>`

```
$ ls -l /etc/rcS.d/  
S55urandom  
S70x11-common
```

- **S**: lanza el script con el argument start
- **K**: lanza el script con el argument stop

Cuando un sistema GNU/Linux arranca, primero se carga el kernel del sistema, después se inicia el primer proceso, denominado init, que es el responsable de ejecutar y activar el resto del sistema, mediante la gestión de los niveles de ejecución (o runlevels).

En el caso del modelo runlevel de SystemV, cuando el proceso init arranca, utiliza un fichero de configuración llamado `/etc/inittab` para decidir el modo de ejecución en el que va a entrar. En este fichero se define el runlevel por defecto (initdefault) en arranque (por instalación en Fedora el 5, en Debian el 2), y una serie de servicios de terminal por activar para atender la entrada del usuario. Después, el sistema, según el runlevel escogido, consulta los ficheros contenidos en `/etc/rcn.d`, donde n es el número asociado al runlevel (nivel escogido), en el que se encuentra una lista de servicios por activar o parar en caso de que arranquemos en el

runlevel, o lo abandonemos. Dentro del directorio encontraremos una serie de script so enlaces a los scripts que controlan el servicio. Cada script posee un nombre relacionado con el servicio, una S o K inicial que indica si es el script para iniciar (S) o matar (K) el servicio, y un número que refleja el orden en que se ejecutarán los servicios.

No todas las distribuciones respetan los estándares.

**(f) Archivo /etc/inittab. ¿Cuál es su finalidad? ¿Qué tipo de información se almacena en él? ¿Cuál es la estructura de la información que en él se almacena?**

Es el archivo de configuración de init. Cuando el sistema se arranca, se verifica si existe un runlevel predeterminado en el archivo /etc/inittab, si no, se debe introducir por medio de la consola del sistema. Después se procede a ejecutar todos los scripts relativos al runlevel especificado.

**(g) Suponga que se encuentra en el runlevel <X>. Indique qué comando(s) ejecutaría para cambiar al runlevel <Y>. ¿Este cambio es permanente? ¿Por qué?**

Con el comando init. No, no es permanente, porque puedo usarlo nuevamente para ir a otro runlevel.

**(h) Scripts RC. ¿Cuál es su finalidad? ¿Dónde se almacenan? Cuando un sistema GNU/Linux arranca o se detiene se ejecutan scripts, indique cómo determina qué script ejecutar ante cada acción. ¿Existe un orden para llamarlos? Justifique.**

Los scripts RC se encargan de cargar o cerrar los servicios necesarios para que el sistema funcione, de acuerdo con el runlevel que se está iniciando. Por ejemplo: lpd(servicio para imprimir), fetchmail (servicio para leer correo-e), sshd (SecureShell, para abrir sesiones remotas de una manera segura), networking (abre las conexiones de red). Todos estos servicios se encuentran en /etc/init.d/. Sin embargo, no todos los servicios se cargan en todos los runlevels.

Los servicios a cargar se encuentran en el directorio /etc/rcX.d/, donde X es el runlevel a cargar. En realidad, en estos directorios no hay más que enlaces simbólicos a /etc/init.d/ (referencias)

### **Nombres de los enlaces simbólicos**

Los nombres en estos directorios tienen una sintaxis bastante concreta. Empiezan por un letra (S o K) seguidos de un número y el nombre del servicio. La letra S significa iniciar (S de start). La letra K significa acabar (K de kill). El número es de dos dígitos, de 00 a 99 e indican el orden en el que se arrancará el servicio

/etc/rc.d/rc cuando entra en un determinado nivel de ejecución realiza las siguientes acciones:

1. Ejecuta, por orden de nombre, todos los scripts que comienzan por K en el directorio correspondiente al nivel, utilizando como argumento para dicho script la opción stop.
2. Ejecuta, por orden de nombre, todos los scripts que comienzan por S en el directorio correspondiente al nivel, utilizando como argumento para dicho script la opción start.

- Los scripts que se ejecutan están en /etc/init.d
- En /etc/rcX.d (donde X = 0..6) hay links a los archivos del /etc/init.d

- Formato de los links:

**[S|K]<orden><nombreScript>**

```
$ ls -l /etc/rcS.d/
S55urandom
S70x11-common
```

- **S:** lanza el script con el argument start
- **K:** lanza el script con el argument stop

**(i) ¿Qué es insserv? ¿Para qué se utiliza? ¿Qué ventajas provee respecto de un arranque tradicional?**

El programa insserv se usa para administrar el orden de los enlaces simbólicos que están en «/etc/rcX.d/», resolviendo las dependencias de forma automática. Mejora la performance del arranque en sistemas multiprocesadores.

Utiliza las cabeceras LSB de los scripts de init.d. Esto permite a cada mantenedor de paquetes especificar en su script de init.d la relación con otros scripts y poder detectar y evitar bucles de dependencias entre scripts así como asegurarse de que todos los scripts se inician en el orden pretendido.

Basicamente es una herramienta de bajo nivel utilizada por update-rc.d que permite el inicio de un sistema instalado script ('secuencia de comandos de arranque') leyendo el encabezado del comentario de la secuencia de comandos, por ejemplo:

```
### BEGIN INIT INFO
# Provides:          boot_facility_1 [ boot_facility_2 ...]
# Required-Start:    boot_facility_1 [ boot_facility_2 ...]
# Required-Stop:     boot_facility_1 [ boot_facility_2 ...]
# Should-Start:      boot_facility_1 [ boot_facility_2 ...]
# Should-Stop:       boot_facility_1 [ boot_facility_2 ...]
# X-Start-Before:    boot_facility_1 [ boot_facility_2 ...]
# X-Stop-After:      boot_facility_1 [ boot_facility_2 ...]
# Default-Start:     run_level_1 [ run_level_2 ...]
# Default-Stop:      run_level_1 [ run_level_2 ...]
# X-Interactive:     true
# Short-Description: single_line_description
# Description:       multiline_description
### END INIT INFO
```

y calcular las dependencias entre todos los guiones. No se recomienda ejecutar insserv directamente a menos que sepa exactamente lo que está haciendo, hacerlo puede hacer que su arranque sistema inoperable. update-rc.d es la interfaz recomendada para administrar scripts de inicio.

<https://manpages.ubuntu.com/manpages/xenial/man8/insserv.8.html>

**(j) ¿Cómo maneja Upstart el proceso de arranque del sistema?**

**(k) Cite las principales diferencias entre SystemV y Upstart.**

El Upstart es un reemplazo basado en eventos, y no en niveles. Los servicios se pueden levantar o desactivaren respuesta a ciertos eventos, y este procedimiento permite por ejemplo manejar el reinicio de servicios que mueren de forma inesperada. Upstart opera asíncronamente. Las tareas y servicios son ejecutados ante eventos (arranque del equipo o inserción de un dispositivo USB) definidos como tareas o Jobs. Los jobs se almacenan en el directorio `/etc/init`. Son scripts en texto plano que definen las acciones a ejecutar. Es compatible con el System V.

Upstart es un reemplazo basado en eventos para el daemon `init`, el método utilizado por varios sistemas operativos Unix-like para realizar tareas durante el arranque del sistema. Upstart trabaja de forma asíncrona supervisando las tareas mientras el sistema esta arrancado. También gestiona las tareas y servicios de inicio cuando el sistema arranca y los detiene cuando el sistema se apaga

La ventaja de usarlo es que no es necesario pedirle a otro paquete que mueva su script a otro número sino que el mismo ya se encarga de los problemas de localización.

Una de las principales diferencias entre System V y Upstart es que el primero trabaja de forma síncrona mientras que Upstart lo hace de forma asíncrona, es decir, no arranca/para un servicio después de otro sino que puede hacerlo en paralelo. También gestiona las tareas y servicios de inicio cuando el sistema arranca y los detiene cuando el sistema se apaga.

El demonio `init` tradicional (SystemV) es estrictamente síncrono, bloqueando futuras tareas hasta que la actual se haya completado. Sus tareas deben ser definidas por adelantado, y solo pueden ser ejecutadas cuando el demonio `init` cambia de estado

**(l) Qué reemplaza a los scripts `rc` de SystemV en Upstart? ¿En que ubicación del filesystem se encuentran?**

En concreto Upstart ignora el archivo `/etc/inittab`, en su lugar proporciona un conjunto integrado de scripts de texto plano de arranque que en principio pueden reemplazar totalmente a `/etc/inittab` y a los scripts de arranque específico del modo de ejecución de SysV, que en vez de encontrarse bajo directorios como `/etc/init.d/rc?.d`, `/etc/rc.d/rc?.d` o `/etc/rc?.d` se encuentran en el directorio `/etc/init`, con nombre `'servicio'.conf`, donde `servicio` es el programa que `init` tratará como un job. Los scripts de Upstart ofrecen más acciones que los de SysV, por ejemplo iniciar un servicio siempre que se conecte un determinado dispositivo de hardware.

Se encuentra en la ruta `/etc/init/*.conf`

**(m) Dado el siguiente job de upstart perteneciente al servicio de base de datos del mysql indique a qué hace referencia cada línea del mismo:**

```
# MySQL Service
description "MySQL Server"
author "info author"
```

**starton (net-device-up** → el trabajo inicia en

**and local-file systems**

**and runlevel [2345])**

**stopon runlevel [016]** → el trabajo para en

[...]

**exec /usr/sbin/mysqld** → Esto especifica lo que se ejecutará para el trabajo.

[...]

**(n) ¿Qué es systemd?**

Es un sistema que centraliza la administración de demonios y librerías del sistema. Mejora el paralelismo del booteo.

**(ñ) ¿A qué hace referencia el concepto de activación de socket en systemd?**

- No todos los servicios que se inician en el booteo se utilizan:
  - impresora
  - servidor en el puerto 80
  - etc.
- Es un mecanismo de iniciación bajo demanda → podemos ofrecer una variedad de servicios sin que realmente estén iniciados
- Cuando el socket recibe una conexión spawnea el servicio y le pasa el socket
- No hay necesidad de definir dependencias entre servicios → se inician todos los sockets en primer medida

**(o) ¿A qué hace referencia el concepto de cgroup?**

- Permite organizar un grupo de procesos en forma jerárquica
- Agrupa conjunto de procesos relacionados (por ejemplo, un servidor web Apache con sus dependientes)
- Tareas que realiza:
  - Tracking mediante subsistema cgroups → no se utiliza el PID → doble *fork* no funciona para escapar de systemd
  - Limitar el uso de recursos
  - etc.

Es una característica del Kernel de Linux que limita, explica y aísla el uso de recursos (CPU, memoria, E / S de disco, red, etc.) de una colección de procesos.

### 3. Usuarios:

**(a) ¿Qué archivos son utilizados en un sistema GNU/Linux para guardar la información**

/home en cuanto a sus “cosas” que quieran hacer, en cuanto a información personal son /etc/passwd o /etc/shadow

**de los usuarios?**

**(b) ¿A qué hacen referencia las siglas UID y GID? ¿Pueden coexistir UIDs iguales en un sistema GNU/Linux? Justifique.**

UID: USER ID

GID: GROUP ID

Si ponemos dos UID iguales a distinto usuario, el S.O. los tomará como una sola cuenta

**(c) ¿Qué es el usuario root? ¿Puede existir más de un usuario con este perfil en GNU/Linux? ¿Cuál es la UID del root?.**

**Es el administrador. Sólo hay un usuario root, pero pueden haber más de un usuario administrador.**

Ser usuario root en un sistema Unix significa tener acceso al directorio raíz, ese donde tenemos instalado todo el sistema operativo. También se le llama ser un Superusuario y en definitiva es tener acceso total al sistema. Algo parecido a tener derechos de administrador en un sistema Windows.

Solo puede haber 1 usuario root pero pueden haber varios admins

UID = 0

Usuario root

- También llamado súper usuario o administrador.
- Su UID (User ID) y GID es 0 (cero).
- Es la única cuenta de usuario con privilegios sobre todo el sistema.
- Acceso total a todos los archivos y directorios con independencia de propietarios y permisos.
- Controla la administración de cuentas de usuarios.
- Ejecuta tareas de mantenimiento del sistema.
- Puede detener el sistema.
- Instala software en el sistema.
- Puede modificar o reconfigurar el kernel, controladores, etc

**(d) Agregue un nuevo usuario llamado iso2017 a su instalación de GNU/Linux, especifique que su home sea creada en /home/iso\_2017, y hágalo miembro del grupo catedra (si no existe, deberá crearlo). Luego, sin iniciar sesión como este usuario cree un archivo en su home personal que le pertenezca. Luego de todo esto, borre el usuario y verifique que no queden registros de él en los archivos de información de los usuarios y grupos.**

```
|root@debian:/# sudo groupadd catedra
```

creo grupo

```
agusnfr@debian:/home$ su
Contraseña:
root@debian:/home# sudo useradd -d /home/iso_2017 -m iso2017
```

-d establezco directorio, con -m creo directorio

```
root@debian:/home# cd /home/iso_2017
root@debian:/home/iso_2017# touch algo.txt
root@debian:/home/iso_2017# ls
algo.txt
root@debian:/home/iso_2017#
```

```
root@debian:/home/iso_2017# sudo usermod -a -G catedra iso2017
root@debian:/home/iso_2017#
```

```
iso2017:x:1001:1002::/home/iso_2017:/bin/sh
root@debian:/#
```

userdel iso2017 ; elimino usuario

rm -r /home/iso\_2017 ; elimino su directorio

o

sudo userdel -r iso\_2017

*Usa el comando userdel para quitar al usuario antiguo:*

*userdel nombre del usuario*

*Opcional: También puedes borrar el directorio principal y la cola de correo de ese usuario con el indicador -r con el comando:*

*userdel -r nombre del usuario*

**(e) Investigue la funcionalidad y parámetros de los siguientes comandos:**

- su: permite usar el intérprete de comandos de otro usuario sin necesidad de cerrar la sesión
- groupadd nombre\_grupo : crea un grupo.
- who: dice el usuario activo en ese momento
- useradd <nombreUsuario>:

Agrega el usuario

Modifica los archivos /etc/passwd

Alternativa → adduser

- passwd <nombreUsuario>:

Asigna o cambia la contraseña del usuario

Modifica el archivo /etc/shadow

- usermod <nombreUsuario>:

- g: modifica grupo de login (Modifica /etc/passwd)
- G: modifica grupos adicionales (Modifica /etc/group)
- d: modifica el directorio home (Modifica /etc/passwd)

- userdel <nombreUsuario>: elimina el usuario
- groupdel <nombreGrupo>: elimina el grupo

#### 4. FileSystem:

##### (a) ¿Cómo son definidos los permisos sobre archivos en un sistema GNU/Linux?

Los permisos están divididos en tres tipos: lectura, escritura y ejecución. Estos permisos pueden ser fijados para tres clases de usuarios: el propietario del archivo o directorio, los integrantes del grupo al que pertenece y todos los demás usuarios.

- Se aplican a directorios y archivos
- Existen 3 tipos de permisos y se basan en una notación octal:

Permiso	Valor	Octal
Lectura	R	4
Escritura	W	2
Ejecución	X	1

- Se aplican sobre los usuarios:
  - Usuario: permisos del dueño → **U**
  - Usuario: permisos del grupo → **G**
  - Usuario: permisos de otros usuario → **O**
- Se utiliza el comando **chmod**:

```
$ chmod 755 /tmp/script
```

##### (b) Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con los permisos en GNU/Linux:

b)i.chmod: es un comando que permite modificar y especificar quien puede manejar el archivo y cómo puede hacerlo.

1.chmod PARAMETROUSUARIO(u,g,o) +(añade permiso para) -(quita permiso para) permiso(r,w,x)

2.ejemplo:

.chmod u+x (da permiso de ejecución al dueño)

.chmod o+r+w (da permiso de lectura/escritura a los otros usuarios)

.chmod g+r-w-x (deja solo permiso de lectura al grupo al que pertenece el archivo)

{ej chmod 755 /tmp/script (chmod modifica los permisos del archivo)}

ii.chown: es un comando que permite cambiar el propietario de un archivo o directorio en GNU. Se puede especificar el nombre de usuario o un GID.



```
.chown nombreUsuario archivo1 [archivo2 archivo3...]
```

```
.chown -R nombreUsuario nombreDirectorio
```

iii.El comando `chgrp` permite cambiar el grupo de usuarios de un archivo o directorio en sistemas tipo UNIX. Cada archivo de Unix tiene un identificador de usuario (UID) y un identificador de grupo (GID), que se corresponden con el usuario y el grupo de quien lo creó.

Normalmente este tipo de tareas de asignación de permisos se puede realizar con el comando `chown` pero `chgrp` maneja una sintaxis mas simple para esta tarea, adicional es un comando de [administrador](#), es decir, únicamente el usuario root puede cambiar el grupo de un archivo o directorio determinado.

La sintaxis de uso de `chgrp` es la siguiente:

```
chgrp [opciones] nuevo_grupo nombre_de_objeto
```

## El comando `chmod`

Sirve para gestionar los permisos de los archivos o directorios del sistema. Cuando usamos este comando debemos tener presentes los tres niveles de gestión de permisos que existen: Lectura, Escritura y Ejecución. Además, tener que tener en cuenta a los tres tipos de usuarios a los que podemos asignar permisos de manera independiente:

- El dueño del archivo (o directorio).
- El grupo al que pertenece.
- Todos los usuarios en general.

Aclarados estos puntos podremos entender la sintaxis del comando `chmod`:

```
chmod [opciones] permisos archivo/directorio/s
```

Los permisos se pueden indicar de dos maneras, por medio de una sintaxis en números octales (base 8) o por medio de caracteres. Explicaremos la base octal, que puede resultar un poco más compleja, pero que resulta más abreviada y universal. En base octal los permisos se expresan con tres dígitos, del 0 al 7. El primer dígito corresponde a los permisos del usuario, el segundo a los del grupo y el tercero a los del resto de usuarios. Por ejemplo, posibles valores de permisos podrían ser 000, 644, 755, 777, etc.

Para entender lo que cada opción significa debemos saber que los números octales tienen su correspondencia con números binarios (ceros y unos). Un cero binario significa que no se tiene permiso y un uno binario que sí se tiene. La tabla de transformación de octal a binario sería la siguiente:

Número	Binario	Lectura (r)	Escritura (w)	Ejecución (x)

0	000	✗	✗	✗
1	001	✗	✗	✓
2	010	✗	✓	✗
3	011	✗	✓	✓
4	100	✓	✗	✗
5	101	✓	✗	✓
6	110	✓	✓	✗
7	111	✓	✓	✓

Así, 0 significa que no hay ningún permiso y 7 que los tiene todos. Ahora, el permiso 000 significa que nadie tiene permiso para hacer nada con el fichero. 777 significa que todo el mundo tiene permisos para hacer cualquier cosa. 700 significa que existen permisos totales para el dueño del archivo, pero no para el resto de los usuarios. Un ejemplo completo de comando sería el siguiente:

## El comando `chown`

Este comando sirve para indicar quién es el dueño y el grupo de un archivo. Su uso es mucho más sencillo de `chmod`. Vamos a verlo con un ejemplo directamente.

```
chown usuariox:grupoy archivo.txt
```

Como se aprecia, se puede especificar el usuario (en este caso *usuariox*) y el grupo (*grupoy*), separados por el carácter `:`. A continuación, se indica el nombre del archivo o directorio afectados por esta nueva configuración.

Igual que `chmod`, se puede especificar que la operación se realice de manera recursiva.

```
chown -R www-data:www-data html
```

Así se especifica que la carpeta de nombre *html* y todo su contenido pertenecerán al usuario y grupo *www-data*.

## Recomendaciones de uso

Como habíamos comentado, estos comandos son muy relevantes en lo que respecta a la seguridad de los ficheros y carpetas del servidor. Queremos finalizar por tanto con unos consejos importantes sobre la seguridad.

- Nunca debemos asignar permisos 777. Aunque a veces pueda parecer que una aplicación web no funciona si no le asignamos esos permisos a una carpeta donde se desea escribir información, el nivel 777 es muy peligroso ya que estamos asignando permisos totales a esos archivos o carpetas. Generalmente, hay que buscar soluciones de permisos más restrictivas y, en cambio, modificar los propietarios o el grupo de los archivos o carpetas.
- Generalmente, trabajamos con 644 para ficheros y 755 para directorios. Esos son los permisos más estándar para los archivos y carpetas del servidor. Esta opción genérica no tiene por qué ser la necesaria para todos los proyectos, pero siempre es una solución bastante apropiada.  
Nuevamente, si nuestra aplicación no funciona bien con este nivel de permisos, conviene estudiar quién se encuentra asignado como propietario y grupo de los archivos y carpetas. Un problema habitual es que el propietario sea root. Esto podría suponer que, cuando desde un lenguaje de programación se intenta acceder al archivo para su escritura, nos arroje un error de acceso. La solución adecuada nunca sería asignar 777 para que la aplicación funcione, sino encontrar el usuario y grupo adecuados para el contenido de la carpeta.

**(c) Al utilizar el comando `chmod` generalmente se utiliza una notación octal asociada para definir permisos. ¿Qué significa esto? ¿A qué hace referencia cada valor?**

Respondida en el punto anterior

**(d) ¿Existe la posibilidad de que algún usuario del sistema pueda acceder a determinado archivo para el cual no posee permisos? Nombrelo, y realice las pruebas correspondientes.**

Si, el usuario root, ya que tiene acceso administrativo al sistema (posee permisos de superusuario). Los usuarios normales no tienen este acceso por razones de seguridad.

Los usuarios con acceso a `sudo` también pueden acceder con o sin permisos ya que este comando `p` permisos de kernel/superusuario

```
agusnfr@debian: /
agusnfr@debian:/$ ls -l
total 65
lrwxrwxrwx 1 root root 7 ago 30 13:49 bin -> usr/bin
drwxr-xr-x 4 root root 1024 ago 30 14:13 boot
drwxr-xr-x 18 root root 3200 sep 5 15:45 dev
drwxr-xr-x 119 root root 12288 sep 5 17:07 etc
drwxr-xr-x 4 root root 4096 sep 5 17:35 home
lrwxrwxrwx 1 root root 31 ago 30 13:54 initrd.img -> boot/initrd.img-5.10.0-17-amd64
lrwxrwxrwx 1 root root 31 ago 30 13:54 initrd.img.old -> boot/initrd.img-5.10.0-16-amd64
lrwxrwxrwx 1 root root 7 ago 30 13:49 lib -> usr/lib
lrwxrwxrwx 1 root root 9 ago 30 13:49 lib32 -> usr/lib32
lrwxrwxrwx 1 root root 9 ago 30 13:49 lib64 -> usr/lib64
lrwxrwxrwx 1 root root 10 ago 30 13:49 libx32 -> usr/libx32
drwx----- 2 root root 16384 ago 30 13:49 lost+found
drwxr-xr-x 3 root root 4096 ago 30 13:49 media
drwxr-xr-x 2 root root 4096 ago 30 13:49 mnt
drwxr-xr-x 2 root root 4096 ago 30 13:49 opt
dr-xr-xr-x 224 root root 0 sep 5 15:45 proc
drwx----- 4 root root 4096 ago 31 20:00 root
drwxr-xr-x 26 root root 640 sep 5 15:47 run
lrwxrwxrwx 1 root root 8 ago 30 13:49 sbin -> usr/sbin
drwxr-xr-x 2 root root 4096 ago 30 13:49 srv
```

Solo tiene permisos el root en home, pero si hago sudo

```
agusnfr@debian:/home$ touch prueba_permiso
touch: no se puede efectuar 'touch' sobre 'prueba_permiso': Permiso denegado
agusnfr@debian:/home$ no se puede
bash: no: orden no encontrada
agusnfr@debian:/home$ sudo touch prueba_permiso
agusnfr@debian:/home$
```

(e) Explique los conceptos de “full path name” y “relative path name”. De ejemplos claros de cada uno de ellos.

“full path name” es la ruta desde el directorio raíz (root) "/" y “relative path name” es la ruta iniciando desde la carpeta desde donde "se esta parado"

Artículo	Descripción
nombre de ruta absoluto	Rastrea la ruta desde el directorio /(raíz). Los nombres de ruta absolutos siempre comienzan con el símbolo de barra inclinada ( / ).
nombre de ruta relativo	Rastrea la ruta desde el directorio actual a través de su padre o sus subdirectorios y archivos.

Un nombre de ruta absoluto representa el nombre completo de un directorio o archivo desde el directorio /(raíz) hacia abajo. Independientemente de dónde esté trabajando en el sistema de archivos, siempre puede encontrar un directorio o archivo especificando su nombre de ruta absoluto. Ruta absoluta los nombres comienzan con una barra inclinada ( / ), el símbolo que representa el directorio raíz. El nombre de ruta /A/D/9 es el nombre de ruta absoluto para 9. La primera barra inclinada ( / )

representa el directorio /(raíz) , que es el lugar de inicio de la búsqueda El resto del nombre de la ruta dirige la búsqueda a A , luego a D y finalmente a 9 .

Pueden existir dos archivos llamados 9 porque los nombres de ruta absolutos a los archivos dan a cada archivo un nombre único dentro del sistema de archivos. Los nombres de ruta /A/D/9 y /C/E/G/9 especifican dos archivos únicos llamados 9 .

A diferencia de los nombres de rutas completas, los nombres de rutas relativas especifican un directorio o archivo basado en el directorio de trabajo actual. Para los nombres de rutas relativas, puede usar la notación punto punto (..) para moverse hacia arriba en la jerarquía del sistema de archivos. El punto punto ( . .) representa el directorio principal. Debido a que los nombres de rutas relativas especifican una ruta que comienza en el directorio actual, no comienzan con una barra inclinada (/). Los nombres de rutas relativas se usan para especificar el nombre de un archivo en el directorio actual o la ruta nombre de un archivo o directorio por encima o por debajo del nivel del directorio actual en el sistema de archivos. Si D es el directorio actual, el nombre de la ruta relativa para acceder a 10 es F/10 . Sin embargo, el nombre de la ruta absoluta siempre es /A/ D/F/10Además, el nombre de la ruta relativa para acceder a 3 es ../../B/3 .

También puede representar el nombre del directorio actual usando la notación punto ( . ) La notación punto ( . ) se usa comúnmente cuando se ejecutan programas que leen el nombre del directorio actual.

**(f) ¿Con qué comando puede determinar en qué directorio se encuentra actualmente? ¿Existe alguna forma de ingresar a su directorio personal sin necesidad de escribir todo el path completo? ¿Podría utilizar la misma idea para acceder a otros directorios? ¿Cómo? Explique con un ejemplo.**

Con pwd.

Comando	Función
cd	vuelve a su directorio de login
cd ~	vuelve también a su directorio de login
cd /	le lleva al directorio raíz del sistema completo
cd /root	le lleva al directorio principal del root, o superusuario, cuenta creada en la instalacion; debe ser el usuario root para accesar este directorio.

Comando	Función
<code>cd /home</code>	lo lleva a su directorio principal, donde los directorios login de usuario son almacenados
<code>cd ..</code>	le traslada a un directorio superior
<code>cd ~otheruser</code>	le lleva al directorio login del usuario <i>otheruser</i> , si <i>otheruser</i> le ha dado permiso
<code>cd /dir1/subdirfoo</code>	sin tener en cuenta en que directorio esta, este recorrido absoluto le llevara directamente a <code>subdirfoo</code> , un subdirectorio de <code>dir1</code>
<code>cd ../../dir3/dir2</code>	este recorrido relativo lo llevara dos directorios mas arriba, luego a <code>dir3</code> , luego al directorio <code>dir2</code> .

**(g) Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el uso del FileSystem:**

- cd: cambia de directorio
- mkdir: crea un directorio
- rmdir: elimina un directorio
- mount: monta un dispositivo
- umount: desmonta un dispositivo
- du: muestra lo que ocupa y el tamaño total de los directorios dentro del directorio donde me encuentre
- df: muestra los sistemas de ficheros montados
- ln: crea enlaces a archivos y crea un fichero que apunta a otro
- ls: lista los archivos y directorios dentro del entorno de trabajo
- pwd: muestra el directorio actual de trabajo
- cp: copia archivos en el directorio indicado
- mv: renombra un conjunto

## 5. Procesos:

**(a) ¿Qué es un proceso? ¿A qué hacen referencia las siglas PID y PPID? ¿Todos los procesos tienen estos atributos en GNU/Linux? Justifique. Indique qué otros atributos tiene un proceso.**

Es un programa en ejecución. Un proceso es una entidad "viva" /se modifica/es dinámico, a diferencia de los programas que son estáticos

PID (Process ID): Es un identificador numérico único para cada proceso.

PPID (Process Parent ID): El identificador del proceso padre.

Todos los procesos tienen estos atributos, además de estos (que son los más importantes, pero no todos) usuario (UID), grupo (GID), prioridad.

con `ps -ejH` puedo ver el PPID

- Mirar todos los procesos que se están ejecutando en el sistema: `ps -ef`
- Mirar todos los procesos que está ejecutando un usuario: `ps -fu usuario`
- Filtrar algunas de las columnas de `ps`: `ps -eopid,tt,user,fname,tmout,f,wchan`

# `ps -flU usuario1`

F S UID PID PPID C PRI NI ADDR SZ WCHAN STIME TTY TIME CMD

4 S usuario1 5083 5082 0 80 0 - 22179 wait 11:45 pts/1 00:00:00 -bash

0 R usuario1 5118 5083 0 80 0 - 22134 - 11:45 pts/1 00:00:00 ps -f

#

**El significado de cada una de las columnas es:**

UID → Propietario del proceso.

PID → ID del proceso.

PPID → ID del proceso padre.

C → Cantidad de recursos de CPU que el proceso ha utilizado recientemente para que el kernel (núcleo del sistema operativo) establezca la prioridad apropiada a cada proceso.

PRI → Prioridad del proceso.

NI → Valor nice. Un valor positivo indica menor tiempo de CPU.

STIME → Hora de comienzo del proceso.

TTY → Terminal asociado al proceso.

TIME → Tiempo de CPU asociado al proceso.

CMD → Comando ejecutado

**(b) Indique qué comandos se podrían utilizar para ver qué procesos están en ejecución en un sistema GNU/Linux.**

Para ver los procesos en sistemas Linux, contamos con el comando 'ps', que listará (de múltiples formas según las opciones que le pasemos) todos los procesos que se encuentran corriendo en nuestro equipo.

ps aux (muestra todos los procesos del sistema)

ps axjf (que mostrará un árbol jerárquico con la ruta del programa al que pertenece el proceso)

Aquí un ejemplo de filtrado sobre ps para obtener únicamente los procesos pertenecientes a bash: ps aux | grep bash

Top es otro gestor de procesos integrado en la mayoría de sistemas Linux. Mientras que ps nos muestra un listado de procesos estático, es decir, nos informa de los procesos, nombres, usuarios o recursos que se están usando en el momento de la petición; top nos da un informe en tiempo real de los mismos.

top -d 5 (Donde 5 es el número de segundos a transcurrir entre cada muestreo)

top -o %CPU (Donde %CPU es el valor por el que vamos a ordenar los procesos )

top -u touchiro (Donde Touchiro es el usuario del cual queremos mostrar los procesos)

Otro gestor de procesos muy interesante y usado es 'htop', que nos mostrará sin salir de la terminal (si es que lo ejecutamos desde ésta...) algo similar a top, pero donde mediante las teclas de función del teclado, accederemos a menús de configuración al estilo de las aplicaciones DOS (qué tiempos...).

Los sistemas Linux vienen con la herramienta KILL instalada, que usaremos para detener los procesos que necesitemos. Por defecto el comando kill envía una señal denominada TERM a un proceso que le pasaremos mediante su PID como argumento. Esta señal TERM pedirá a dicho proceso que termine, permitiéndole gestionar su función de cierre, completando las tareas necesarias y limpiando la información que ha cargado en memoria.

kill [PID del proceso]

### **(c) ¿Qué significa que un proceso se está ejecutando en Background? ¿Y en Foreground?**

El estado background se refiere a que, al ser activado desde una terminal, el proceso se ejecuta de manera independiente a la terminal sin "amarrarla" durante el tiempo de proceso.

El estado foreground en cambio, "amarra" la terminal al proceso dejándola sin capacidad de correr otra tarea en esa terminal mientras el proceso se ejecuta.

Cuando un proceso está en ejecución sin que sea mostrado en la terminal se dice que se está ejecutando en el background. Si se muestra la ejecución del comando dentro de la terminal se dice que está en el foreground.

### **(d) ¿Cómo puedo hacer para ejecutar un proceso en Background? ¿Como puedo hacer para pasar un proceso de background a foreground y viceversa?**

En lugar de hacer CTRL + C para detener el proceso, con CTRL + Z es enviar el proceso a segundo plano, y aunque está en segundo plano, no está trabajando, está detenido. Ahora ya se puede seguir usando la terminal. Para saber los procesos que están en segundo plano se utiliza el comando jobs. Para iniciar un proceso directamente en segundo plano solo se tiene que añadir & al final de la instrucción.



Para traerlo de background primero se utiliza jobs para saber el número de proceso, y posteriormente fg % seguido del número de proceso (ID).

Los comandos **fg** y **bg** permiten traer un proceso a primer plano o enviarlo a segundo plano respectivamente. Su uso es tan sencillo, como ejecutar la instrucción seguido de % y el número del proceso que obtenemos de jobs.

Así si se quisiera traer el proceso 2 a primer plano, tan solo se tiene que ejecutar fg %2. Pero una vez en primer plano ¿cómo se devuelve a segundo plano? Utilizando el atajo de teclado Ctrl+Z. Sin embargo al utilizar el atajo de teclado se ha detenido, y esa no era la intención, se quería que continuara. Para ello, simplemente se utiliza bg %2 y el proceso que se pondrá en marcha de nuevo.

Así la operativa ha sido,

- traer el proceso a primer plano utilizando fg %2
- devolver el proceso a segundo plano con el atajo de teclado Ctrl+Z
- poner el proceso en marcha una vez que está en segundo plano con bg %2.

#### **(e) Pipe ( | ). ¿Cuál es su finalidad? Cite ejemplos de su utilización.**

El “|” nos permite comunicar dos procesos por medio de un pipe desde la shell

El pipe conecta stdout (salida estándar) del primer comando con la stdin (entrada estándar) del segundo.

- Por ejemplo:
  - \$ ls | more
    - Se ejecuta el comando ls y la salida del mismo, es enviada como entrada del comando more

Se pueden anidar tantos pipes como se deseen

*-Grep busca una palabra o patrón en un lugar, y devuelve todas las coincidencias.*

#### **(f) Redirección. ¿Qué tipo de redirecciones existen? ¿Cuál es su finalidad? Cite ejemplos de utilización.**

En GNU/Linux, al final todo es tratado como si fuera un fichero y como tal, tenemos descriptores de fichero para

aquellos puntos donde queramos acceder. Hay unos descriptores de fichero por defecto:

- Entrada estándar (normalmente el teclado).
- 1: Salida estándar (normalmente la consola).
- 2: Salida de error.

Para redirigir las salidas utilizaremos el descriptor de fichero seguido del símbolo '>' (o < si redirigimos la entrada hacia un comando, pero eso todavía no lo explicaremos). Veamos unos ejemplos:

\$ ls 1 >fichero

Guarda la salida de ls 1 en fichero. Si no existe lo crea, y si existe lo sobrescribe.

```
$ ls 1>>fichero
```

Añade la salida del comando a fichero. Si no existe lo crea, y si existe, lo añade al final.

```
$ ls 2>fichero
```

Si hay algún error, lo guarda en fichero (podría salir un error si no tuviéramos permiso de lectura en el directorio).

Es importante ver que si no se especifica el descriptor de fichero se asume que se redirige la salida estándar. En el caso del operador < se redirige la entrada estándar, es decir, el contenido del fichero que especificáramos, se pararía como parámetro al comando.

Si quisiéramos redirigir todas las salidas a la vez hacia un mismo fichero, podríamos utilizar >&. Además, con el carácter & podemos redirigir salidas de un tipo hacia otras, por ejemplo, si quisiéramos redirigir la salida de error hacia la salida estándar podríamos indicarlo con: 2>&1. Es importante tener en cuenta que el orden de las redirecciones es significativo: se ejecutarán de izquierda a derecha.

#### **(g) Comando kill. ¿Cuál es su funcionalidad? Cite ejemplos.**

Kill es un comando unix/linux que permite enviar señales a uno o varios procesos del sistema a través de la shell (bash, ksh, etc). Las señales más utilizadas suelen ser la de matar un proceso (9 ó SIGKILL), pararlo (TERM) o reiniciarlo (1 ó HUP) pero hay muchas más que pueden ser útiles en ocasiones. El listado completo de señales disponibles puede visualizarse ejecutando kill -l

#### **(h) Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el manejo de procesos en GNU/Linux. Además, compárelos entre ellos:**

ps: Muestra información de los procesos activos.

ps [OPCIONES]

Por defecto muestra:

- **PID** : Id del Proceso.
- **TTY** : Terminal.
- **TIME** : Tiempo de ejecución.
- **CMD** : Comando.

Opciones

- **a** : Muestra todos los procesos asociados a un TTY.
- **-e / -A** : Muestra todos los procesos.
- **x** : Muestra los no asociados.
- **-f** : Muestra el formato largo:
  - **UID** : Usuario que lo ejecutó.
  - **PPID** : Id del proceso padre.
  - **C** : Uso del procesador.

- **STIME** : Inicio de ejecución.
- **u** : Orientado al usuario:
  - **USER**
  - **% CPU** : uso de procesador.
  - **% MEM** : uso de memoria.
  - **VSZ** : Memoria virtual.
  - **RSS** : Memoria física.
  - **STAT** : Estado.
  - **START** : Iniciado.

Significados de los estados STAT :

- **S** : Esperando (Sleep).
- **R** : Ejecutando (Running)
- **D** : Esperando entrada/salida.
- **T** : Pausa.
- **Z** : No responde (Zombie)

Información adicional de los STAT:

- **s** : Proceso padre
- **l** : Proceso con hilos.
- **+** : En primer plano.

**ps tree**: muestra en forma de árbol los procesos en ejecución.

Con el parámetro “-a” , nos muestra la línea de comandos utilizada. Por ejemplo, si el comando utiliza la ruta a un fichero de configuración.

Por defecto se inhabilita la visualización en el árbol de los nombres repetidos. Para evitar esto, podemos utilizar el parámetro “-c”

Si nos interesa podemos ver el árbol de un proceso específico, de la siguiente manera:

**ps tree -H [PID]**

**top**: muestra en tiempo real información del sistema. (Por defecto cada 3 segundos)

**\$ top [-d SEGUNDOS] [-u USUARIO]**

**htop**: con htop podremos obtener un resultado más amigable, no esta instalado por defecto se ha de instalar mediante **\$ apt-get install htop ...** una vez instalado ...

**\$ htop**

killall: mata todos los procesos con un nombre concreto o de un usuario o ambos.

```
$ killall [-u USUARIO] [-SENAL] [NOMBRE_PROCESO]
```

Ejemplos de uso:

- Matar todos los procesos con el nombre de "sleep"

```
$ killall sleep
```

- Matar todos los procesos del usuario "Pepito"

```
$ killall -u Pepito
```

- Matar todos los procesos del usuario "Pepito" que se llamen "sleep"

```
$ killall -u Pepito sleep
```

nice: ejecuta un comando con una prioridad distintas ala de por defecto. Solo los usuarios root pueden establecer prioridades urgentes (negativos).

```
$ nice -n NUMERO_PRIORIDAD COMANDO
```

renice: cambia la prioridad de un proceso ejecutándose. No se puede aumentar la urgencia.

```
$ renice -n NUMERO_PRIORIDAD COMANDO
```

\$ nice -10 named (esto bajaría la prioridad de named en -10 unidades (si estaba en -10, pasara a -20) MENOS GENTIL = MAS PRIORIDAD

## **6. Otros comandos de Linux (Indique funcionalidad y parámetros):**

**(a) ¿A qué hace referencia el concepto de empaquetar archivos en GNU/Linux?**

- Empaquetar: Es agrupar en un solo fichero varios ficheros y/o directorios.
- Comprimir: Es reducir el tamaño de un fichero a través del uso de un algoritmo de compresión.

Empaquetar

En linux contamos con el comando tar, que nos permite realizar el proceso de empaquetación, su sintaxis es la siguiente:

```
tar [opciones] [nombre_fichero_tar] [directorio_origen]
```

Las opciones más utilizadas son (la versión con un guion es la corta y con dos guiones la larga, pero hacen lo mismo):

-c``--create: Crea un nuevo archivo.

-x``--extract: Extrae ficheros de un archivo.

-v``--verbose: Lista detalladamente los ficheros procesados.

-f``\$&fichero\$&: Empaqueta o desempaqueta en o hacia un fichero.

-t``--list: Lista los contenidos de un archivo.

Algunos ejemplos de uso son los siguientes:

- Crear un archivo tar llamado edteam.tar con los archivos del directorio cursos.

```
tar -cf edteam.tar cursos
```

- Crear un archivo tar llamado edteam.tar con los archivos del directorio cursos mostrando el detalle de los ficheros procesados.

```
tar -cvf edteam.tar cursos
```

- Ver el contenido del archivo edteam.tar

```
tar -tf edteam.tar
```

### Comprimir

Los comandos gzip y gunzip permiten comprimir y descomprimir ficheros respectivamente, su sintaxis básica es:

```
gzip [archivo_a_comprimir]
```

```
gunzip [archivo_a_descomprimir]
```

Por ejemplo, para comprimir el archivo edteam.tar usaremos:

```
gzip edteam.tar
```

El comando anterior generará el archivo edteam.tar.gz, de manera que para descomprimir dicho archivo usaremos:

```
gunzip edteam.tar.gz
```

También podemos realizar el proceso de empaquetación y compresión a través de una sola instrucción, agregando la opción -z al comando tar.

Veamos algunos ejemplos:

- Crear un archivo empaquetado y comprimido llamado edteam.tar.gz con los archivos del directorio cursos mostrando el detalle de los ficheros procesados.

```
tar -cvzf edteam.tar.gz cursos
```

- Desempaqueta y descomprime el archivo edteam.tar.gz mostrando el detalle de los ficheros procesados.  
tar -xvzf edteam.tar.gz

### Comprimir con zip

Además de gzip y unzip, podemos comprimir y descomprimir a través de zip y unzip respectivamente, este formato de compresión es el más utilizado en sistemas operativos Windows.

Su sintaxis básica es la siguiente:

```
zip -r [nombre_fichero_zip] [directorio_a_comprimir]
```

```
unzip [nombre_fichero_zip]
```

- Crear un archivo comprimido llamado edteam.zip con los archivos del directorio cursos:  
zip -r edteam.zip cursos
- Descomprimir el archivo edteam.zip  
unzip edteam.zip

**(b) Seleccione 4 archivos dentro de algún directorio al que tenga permiso y sume el tamaño de cada uno de estos archivos. Cree un archivo empaquetado conteniendo estos 4 archivos y compare los tamaños de los mismos. ¿Qué característica nota?**

<b>Empaquetar archivos:</b>	tar cvf nombre.tar directorio_o_nombres_archivos
<b>Desempaquetar archivos:</b>	tar xvf nombre.tar
<b>Comprimir archivos:</b>	gzip nombre_archivo_a_comprimir
<b>Descomprimir archivos:</b>	gzip -x nombre_archivo_a_descomprimir

```

agusnfr@debian:~$ cd pruebaGRal/
agusnfr@debian:~/pruebaGRal$ touch prueba1
agusnfr@debian:~/pruebaGRal$ touch prueba2
agusnfr@debian:~/pruebaGRal$ touch prueba3
agusnfr@debian:~/pruebaGRal$ touch prueba4
agusnfr@debian:~/pruebaGRal$ ls
prueba1 prueba2 prueba3 prueba4
agusnfr@debian:~/pruebaGRal$ ^Cr cvf prueba.tar prueba GRal
agusnfr@debian:~/pruebaGRal$ cd ..
agusnfr@debian:~$ tar cvf prueba.tar pruebaGRal
pruebaGRal/
pruebaGRal/prueba1
pruebaGRal/prueba4
pruebaGRal/prueba3
pruebaGRal/prueba2

```

```

agusnfr@debian:~$ du -sh *
376K   Descargas
16K    Documentos
4,0K   Escritorio
1,6M   Imágenes
4,0K   Música
4,0K   Plantillas
4,0K   pruebaGRal
12K    prueba.tar
4,0K   Público
4,0K   Vídeos

```

**(c) ¿Qué acciones debe llevar a cabo para comprimir 4 archivos en uno solo? Indique la secuencia de comandos ejecutados**

```

agusnfr@debian:~$ tar -cvzf prueba2.tar.gz pruebaGRal
pruebaGRal/
pruebaGRal/prueba1
pruebaGRal/prueba4
pruebaGRal/prueba3
pruebaGRal/prueba2

```

```

agusnfr@debian:~$ du -sh *
376K   Descargas
16K    Documentos
4,0K   Escritorio
1,6M   Imágenes
4,0K   Música
4,0K   Plantillas
4,0K   prueba2.tar.gz
4,0K   pruebaGRal
12K    prueba.tar
4,0K   Público
4,0K   Vídeos

```

Pesa lo mismo

**(d) ¿Pueden comprimirse un conjunto de archivos utilizando un único comando?**

No, se deben comprimir de a uno

**(e) Investigue la funcionalidad de los siguientes comandos:**

tar : permite empaquetar varios archivos en uno solo, sin comprimirlos. También sirva para desempaquetar.

Ej: tar -vcf nombre\_archivo.tar nombre\_carpeta\_a\_empaquetar

Ej: tar -vxf mi\_archivo.tar

grep : muestra líneas que concuerdan con un patrón.

gzip : comprime o expande ficheros.

zgrep : busca una expresión regular en ficheros posiblemente comprimidos.

wc : sirve para contar líneas, palabras y caracteres que contiene un archivo.

7. Indique qué acción realiza cada uno de los comandos indicados a continuación considerando su orden. Suponga que se ejecutan desde un usuario que no es root ni pertenece al grupo de

root. (Asuma que se encuentra posicionado en el directorio de trabajo del usuario con el que se logueó). En caso de no poder ejecutarse el comando, indique la razón:

ls -l > prueba → se guarda la salida de ls -l en prueba

ps > PRUEBA → se guarda la salida de ps en PRUEBA

chmod 710 prueba → le da todos los permisos al creador de prueba, permisos de ejecución al grupo de prueba y ningún permiso a otros

chown root : root PRUEBA → cambia el propietario de PRUEBA a root y el grupo de PRUEBA a root, no puede ser realizado por un no superusuario

chmod 777 PRUEBA → le da todos los permisos al creador de PRUEBA, al grupo de PRUEBA y a otros usuarios.

chmod 700 /etc/passwd → le da todos los permisos al creador del directorio /etc/passwd y ninguno al grupo y a otros usuarios. No está permitida para un usuario no root ya que contiene los archivos que generalmente se utilizan en la administración del sistema.

passwd root → intenta cambiar la contraseña al root, un usuario no root no puede hacerlo.

rm PRUEBA → se elimina el archivo PRUEBA

man /etc/shadow → intenta desplegar manual de /etc/shadow pero no es posible dado que no es un comando, sino una ruta

find / -name \*.conf → lista todos los archivos cuyos nombres son name y terminan en .conf empezando la búsqueda en "/"

usermod root -d /home/newroot -L → cambia la ruta del root a "/home/newroot" y bloquea la contraseña, un usuario no puede cambiarle el home al root

cd /root → intenta acceder al directorio root

rm \* → borra todos los archivos del directorio

cd /etc → se accede al directorio /etc

cp \* /home -R → intenta copiar todo lo que está en el directorio donde se está parado a home. El -R indica que se haga lo mismo con los archivos y subcarpetas internos

shutdown → apaga la pc

**8. Indique qué comando sería necesario ejecutar para realizar cada una de las siguientes acciones:**

**(a) Terminar el proceso con PID 23.**

kill SIGKILL 23 o kill -9 63772. EL COMANDO KILL ENVIA SEÑALES, LA SEÑAL SIGKILL ES PARA TERMINARLO

**(b) Terminar el proceso llamado init. ¿Qué resultados obtuvo?**

Kill -9 1, no se puede matar al proceso init

**(c) Buscar todos los archivos de usuarios en los que su nombre contiene la cadena ".conf"**



find / -name \*.conf

**(d) Guardar una lista de procesos en ejecución el archivo /home/<su nombre de usuario>/procesos**

ps > /home/agusnfr/procesos

**(e) Cambiar los permisos del archivo /home/<su nombre de usuario>/xxxx a:**

- **Usuario:** Lectura, escritura, ejecución
- **Grupo:** Lectura, ejecución
- **Otros:** ejecución

chmod 751 /home/agusnfr/xxxx

**(f) Cambiar los permisos del archivo /home/<su nombre de usuario>/yyyy a:**

**Usuario:** Lectura, escritura.

**Grupo:** Lectura, ejecución

**Otros:** Ninguno

chmod 650 /home/agusnfr/yyyy

**(g) Borrar todos los archivos del directorio /tmp**

rm /tmp/\*

**(h) Cambiar el propietario del archivo /opt/isodata al usuario iso2010**

chown iso2010 /opt/isodata

**(i) Guardar en el archivo /home/<su nombre de usuario>/donde el directorio donde me encuentre en este momento, en caso de que el archivo exista no se debe eliminar su contenido anterior**

pwd>>/home/agusnfr/donde

**9. Indique qué comando sería necesario ejecutar para realizar cada una de las siguientes acciones:**

**(a) Ingrese al sistema como usuario "root"**

su

**(b) Cree un usuario. Elija como nombre, por convención, la primer letra de su nombre seguida de su apellido. Asígnele una contraseña de acceso.**

useradd -m ARojas

passwd ARojas

ingreso contraseña y confirmo

**(c) ¿Qué archivos fueron modificados luego de crear el usuario y qué directorios se crearon?**

Se modifico /etc/shadow /etc/group /etc/passwd y se creo el directorio /home/ARojas

**(d) Crear un directorio en /tmp llamado cursada2017**

```
mkdir /tmp/cursada2017
```

**(e) Copiar todos los archivos de /var/log al directorio antes creado.**

```
cp * /tmp/cursada2017 -R (estoy parada en /var/log)
```

**(f) Para el directorio antes creado (y los archivos y subdirectorios contenidos en él) cambiar el propietario y grupo al usuario creado y grupo users.**

```
chown ARojas:users /tmp/cursada2017 -R
```

**(g) Agregue permiso total al dueño, de escritura al grupo y escritura y ejecución a todos los demás usuarios para todos los archivos dentro de un directorio en forma recursiva.**

```
chmod 723 /tmp/cursada2017 -R
```

**(h) Acceda a otra terminal virtual para loguearse con el usuario antes creado.**



**(i) Una vez logueado con el usuario antes creado, averigüe cuál es el nombre de su terminal.**

```
whoami
```

**(j) Verifique la cantidad de procesos activos que hay en el sistema.**

```
ps
```

**(k) Verifiqué la cantidad de usuarios conectados al sistema.**

```
who
```

**(l) Vuelva a la terminal del usuario root, y envíele un mensaje al usuario anteriormente creado, avisándole que el sistema va a ser apagado.**

```
Write ARojas tty3
```

Enviomensajes

Ctrl + D

**(m) Apague el sistema.**

Shutdown

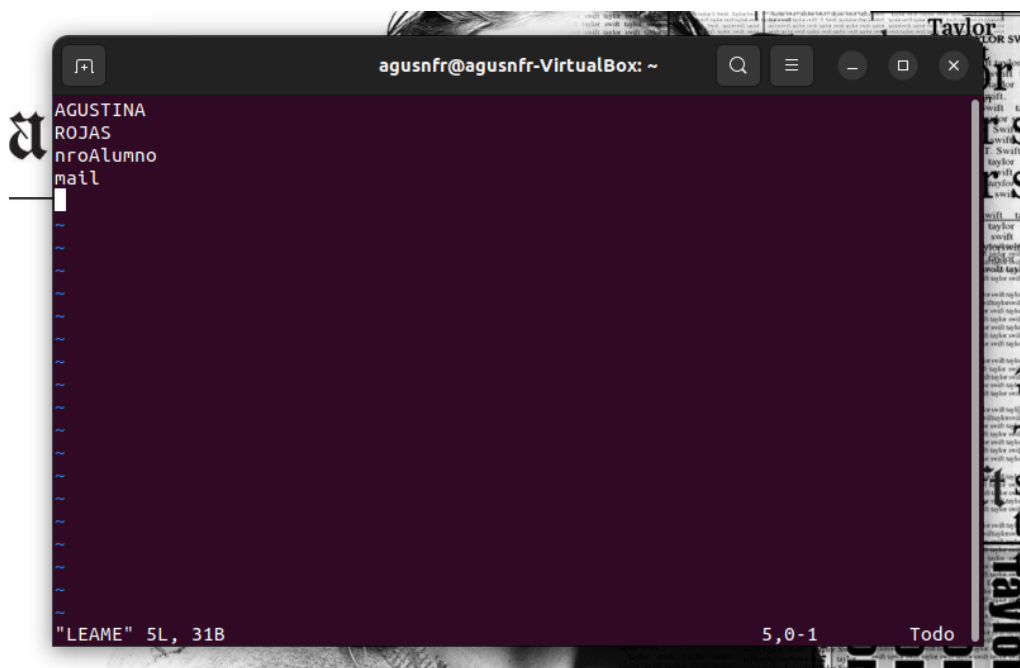
**10. Indique qué comando sería necesario ejecutar para realizar cada una de las siguientes acciones:**

**(a) Cree un directorio cuyo nombre sea su número de legajo e ingrese a él.**

mkdir legajo

cd legajo

**(b) Cree un archivo utilizando el editor de textos vi, e introduzca su información personal: Nombre, Apellido, Número de alumno y dirección de correo electrónico. El archivo debe llamarse "LEAME".**



ESC y :wq! Para salir

I para insertar

V visual

**(c) Cambie los permisos del archivo LEAME, de manera que se puedan ver reflejados los siguientes permisos:**

**Dueño:** ningún permiso

**Grupo:** permiso de ejecución

**Otros:** todos los permisos

(d) Vaya al directorio /etc y verifique su contenido. Cree un archivo dentro de su directorio personal cuyo nombre sea leame donde el contenido del mismo sea el listado de todos los archivos y directorios contenidos en /etc. ¿Cuál es la razón por la cual puede crear este archivo si ya existe un archivo llamado “LEAME” en este directorio?.

chmod 071 LEAME

```
agusnfr@agusnfr-VirtualBox:~$ ls -l
total 44
drwxr-xr-x 2 agusnfr agusnfr 4096 sep  7 21:41 Descargas
drwxr-xr-x 2 agusnfr agusnfr 4096 sep  7 19:31 Documentos
drwxr-xr-x 2 agusnfr agusnfr 4096 sep  8 11:45 Escritorio
drwxr-xr-x 2 agusnfr agusnfr 4096 sep  7 19:31 Imágenes
-----xrw 1 agusnfr agusnfr   30 sep  8 11:54 LEAME
drwxrwxr-x 2 agusnfr agusnfr 4096 sep  8 11:46 legajo
drwxr-xr-x 2 agusnfr agusnfr 4096 sep  7 19:31 Música
drwxr-xr-x 2 agusnfr agusnfr 4096 sep  7 19:31 Plantillas
drwxr-xr-x 2 agusnfr agusnfr 4096 sep  7 19:31 Público
drwx----- 5 agusnfr agusnfr 4096 sep  7 21:42 snap
drwxr-xr-x 2 agusnfr agusnfr 4096 sep  7 19:31 Videos
agusnfr@agusnfr-VirtualBox:~$
```

(e) ¿Qué comando utilizaría y de qué manera si tuviera que localizar un archivo dentro del filesystem? ¿Y si tuviera que localizar varios archivos con características similares? Explique el concepto teórico y ejemplifique.

find / -(i)name “nombre” → un archivo en particular

ej find / -name my-file → busca archivo llamado my-file

ej find / -iname my-file → busca archivo llamado my-file sin distinguir mayúsculas de minúsculas

find / -not -name my-file → busca archivo sin my-file en el nombre

find / -name “\*.txt” → características similares

/ a partir del directorio raíz

. a partir de donde estoy parado

~ directorio personal

Búsqueda por tipo:

d – directorio o carpeta

f – archivo normal

l – enlace simbólico

c – dispositivos de caracteres

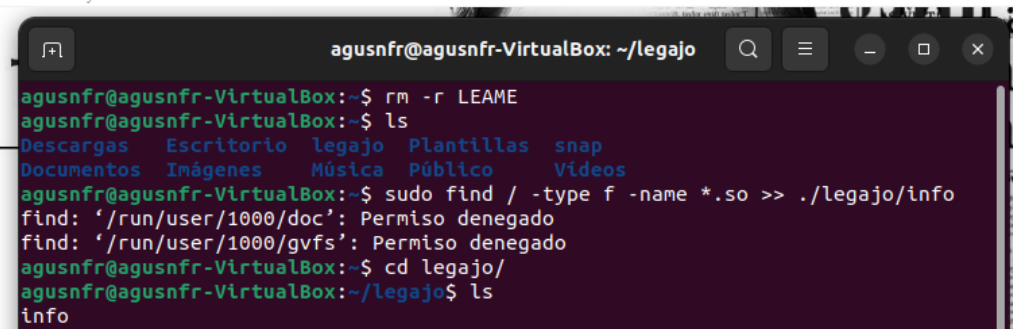
b – dispositivos de bloque

find / -type d → todos los directorios en mi sistema de archivos

find / -type f -name my-file → esto buscará archivos llamados my-file, excluyendo directorios o enlaces.

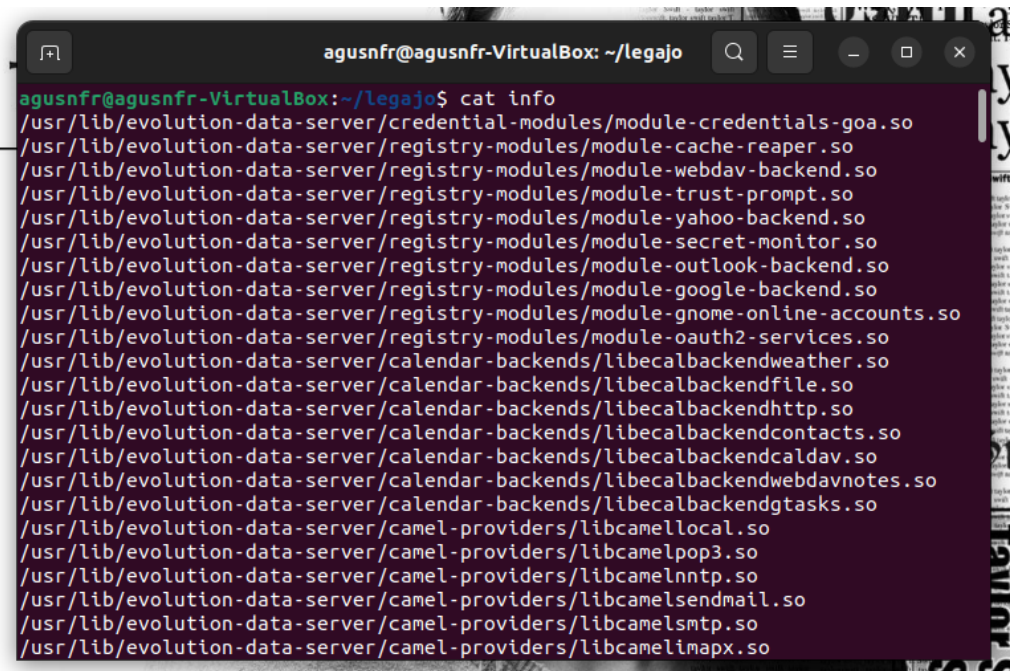
find / -name "LEAME"

(f) Utilizando los conceptos aprendidos en el punto e), busque todos los archivos cuya extensión sea .so y almacene el resultado de esta búsqueda en un archivo dentro del directorio creado en a). El archivo deberá llamarse ejercicio\_f".



```
agusnfr@agusnfr-VirtualBox: ~/legajo
agusnfr@agusnfr-VirtualBox:~$ rm -r LEAME
agusnfr@agusnfr-VirtualBox:~$ ls
Descargas  Escritorio  legajo     Plantillas  snap
Documentos Imágenes   Música     Público     Videos
agusnfr@agusnfr-VirtualBox:~$ sudo find / -type f -name *.so >> ./legajo/info
find: '/run/user/1000/doc': Permiso denegado
find: '/run/user/1000/gvfs': Permiso denegado
agusnfr@agusnfr-VirtualBox:~$ cd legajo/
agusnfr@agusnfr-VirtualBox:~/legajo$ ls
info
```

Supongamos que info es ejercicio\_f me olvide de ponerle ese nombre Xd



```
agusnfr@agusnfr-VirtualBox: ~/legajo
agusnfr@agusnfr-VirtualBox:~/legajo$ cat info
/usr/lib/evolution-data-server/credential-modules/module-credentials-goa.so
/usr/lib/evolution-data-server/registry-modules/module-cache-reaper.so
/usr/lib/evolution-data-server/registry-modules/module-webdav-backend.so
/usr/lib/evolution-data-server/registry-modules/module-trust-prompt.so
/usr/lib/evolution-data-server/registry-modules/module-yahoo-backend.so
/usr/lib/evolution-data-server/registry-modules/module-secret-monitor.so
/usr/lib/evolution-data-server/registry-modules/module-outlook-backend.so
/usr/lib/evolution-data-server/registry-modules/module-google-backend.so
/usr/lib/evolution-data-server/registry-modules/module-gnome-online-accounts.so
/usr/lib/evolution-data-server/registry-modules/module-oauth2-services.so
/usr/lib/evolution-data-server/calendar-backends/libecalbackendweather.so
/usr/lib/evolution-data-server/calendar-backends/libecalbackendfile.so
/usr/lib/evolution-data-server/calendar-backends/libecalbackendhttp.so
/usr/lib/evolution-data-server/calendar-backends/libecalbackendcontacts.so
/usr/lib/evolution-data-server/calendar-backends/libecalbackendcaldav.so
/usr/lib/evolution-data-server/calendar-backends/libecalbackendwebdavnotes.so
/usr/lib/evolution-data-server/calendar-backends/libecalbackendgtasks.so
/usr/lib/evolution-data-server/camel-providers/libcamellocal.so
/usr/lib/evolution-data-server/camel-providers/libcamelpop3.so
/usr/lib/evolution-data-server/camel-providers/libcamelntp.so
/usr/lib/evolution-data-server/camel-providers/libcamelsendmail.so
/usr/lib/evolution-data-server/camel-providers/libcamelsmtp.so
/usr/lib/evolution-data-server/camel-providers/libcamelmapx.so
```

11. Indique qué acción realiza cada uno de los comandos indicados a continuación considerando su orden. Suponga que se ejecutan desde un usuario que no es root ni pertenece al grupo de root. (Asuma que se encuentra posicionado en el directorio de trabajo del usuario con el que se logueó). En caso de no poder ejecutarse el comando indique la razón:

mkdir iso → crea directorio iso

cd . / iso ; ps > f0 → accede a la carpeta iso y se guarda la información de los procesos activos en f0 (que se crea dentro del directorio iso)

ls > f1 → guarda en f1 todos los archivos del directorio actual

cd / → accede al directorio raíz

echo \$HOME → imprime en pantalla (ECHO) el directorio del usuario que está ejecutando

ls -l \$> \$HOME/iso/ls → guarda en el archivo ls ubicado en <directorio del usuario que esta ejecutando el comando>/iso/ls la salida de ls -l

cd \$HOME; mkdir f2 → accede al directorio del usuario que esta ejecutando el comando y crea el directorio f2

ls -ld f2 → lista los datos del directorio f2

chmod 341 f2 → da permisos de escritura y ejecución al creado de f2, de lectura al grupo y de ejecución a otros usuarios

touch dir → crea archivo llamado dir

cd f2 → accede al directorio f2

cd ~/iso → accede a directorio iso ~ es el directorio del usuario que esta ejecutando el comando

pwd >f3 → guarda en f3 la ruta en donde se esta parado

ps | grep 'ps' | wc -l >> . / f2/f3 guarda en el archivo f3 (como se utiliza .. se tiene que realizar el comando en un directorio que este dentro del cual se ubica f2) las veces que esta la palabra "ps" en la lista de procesos

grep ps → busca una palabra o patrón en un lugar, y devuelve todas las coincidencias.

wc -l → cuenta lineas

chmod 700 . /f2 ; cd . . → cambia los permisos de f2 a todos los permisos el creado, ninguno el grupo, ninguno el resto y accede al directorio padre de donde se esta parado en el momento que se ejecuta el comando (que el directorio obviamente en el que se esta parado debe ser uno que tenga como padre a \$HOME, ya que no se podría ejecutar el primer comando).

find . -name etc/passwd busca desde la ruta en la que se esta posicionado archivos que tengan de nombre etc/passwd → tendría que ser wholenam porque name solo sirve para nombres base y etc/passwd contiene separadores de directorios, si se utiliza name no se va a encontrarnada.

find / -name etc/passwd busca desde la raíz en la que se esta posicionado archivos que tengan de nombre etc/passwd → tendría que ser wholenam porque name solo sirve para nombres base y etc/passwd contiene separadores de directorios, si se utiliza name no se va a encontrar nada. Y falta / antes del etc

mkdir ejercicio5 → crea carpeta ejercicio5

**(a) Inicie 2 sesiones utilizando su nombre de usuario y contraseña. En una sesión vaya siguiendo paso a paso las órdenes que se encuentran escritas en el cuadro superior. En la**

otra sesión, cree utilizando algún editor de textos un archivo que se llame "ejercicio10\_explicacion" dentro del directorio creado en el ejercicio 9.a) y, para cada una de las órdenes que ejecute en la otra sesión, realice una breve explicación de los resultados obtenidos.

(b) Complete en el cuadro superior los comandos 19 y 20, de manera tal que realicen la siguiente acción:

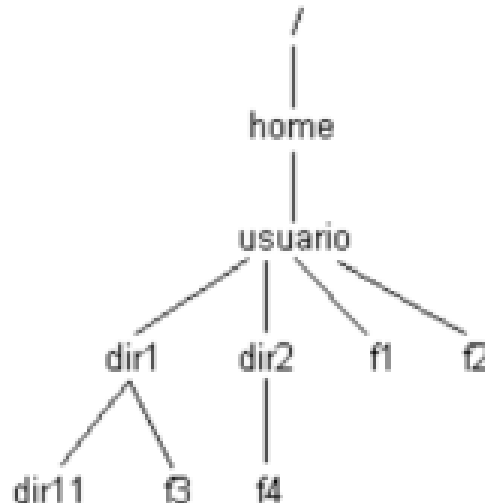
19: Copiar el directorio iso y todo su contenido al directorio creado en el inciso 9.a).

```
cp /home/agusnfr/iso /home/ARojas -R
```

20: Copiar el resto de los archivos y directorios que se crearon en este ejercicio al directorio creado en el ejercicio 9.a).

```
cp -n* /home/ARojas -R
```

(c) Ejecute las órdenes 19 y 20 y coméntelas en el archivo creado en el inciso a).



12. Cree una estructura desde el directorio /home que incluya varios directorios, subdirectorios y archivos, según el esquema siguiente. Asuma que "usuario" indica cuál es su nombre de usuario. Además deberá tener en cuenta que dirX hace referencia a directorios y fX hace referencia a archivos:

(a) Utilizando la estructura de directorios anteriormente creada, indique que comandos son necesarios para realizar las siguientes acciones:

```
agusnfr@agusnfr-VirtualBox:~$ mkdir dir1
agusnfr@agusnfr-VirtualBox:~$ mkdir dir2
agusnfr@agusnfr-VirtualBox:~$ touch f1
agusnfr@agusnfr-VirtualBox:~$ touch f2
agusnfr@agusnfr-VirtualBox:~$ cd dir1
agusnfr@agusnfr-VirtualBox:~/dir1$ mkdir dir11
agusnfr@agusnfr-VirtualBox:~/dir1$ touch f3
agusnfr@agusnfr-VirtualBox:~/dir1$ cd ..
agusnfr@agusnfr-VirtualBox:~$ cd dir2
agusnfr@agusnfr-VirtualBox:~/dir2$ touch f4
agusnfr@agusnfr-VirtualBox:~/dir2$
```

- Mueva el archivo "f3" al directorio de trabajo /home/usuario.

```
agusnfr@agusnfr-VirtualBox:~$ cd dir1
agusnfr@agusnfr-VirtualBox:~/dir1$ mv /home/agusnfr/dir1/f3 /home/agusnfr
agusnfr@agusnfr-VirtualBox:~/dir1$ ls
dir11
agusnfr@agusnfr-VirtualBox:~/dir1$ ls /home/agusnfr
Descargas  Documentos  f2      iso      Plantillas  snap
dir1       Escritorio  f3      legajo   prueba      Videos
dir2       f1          Imágenes Música    Público
agusnfr@agusnfr-VirtualBox:~/dir1$
```

- Copie el archivo "f4" en el directorio "dir11".

```
bash: cd: dir2: No existe el archivo o el directorio
agusnfr@agusnfr-VirtualBox:~/dir1$ cd ../dir2
agusnfr@agusnfr-VirtualBox:~/dir2$ cp f4 ../dir1
agusnfr@agusnfr-VirtualBox:~/dir2$ cd ../dir1
agusnfr@agusnfr-VirtualBox:~/dir1$ ls
dir11  f4
agusnfr@agusnfr-VirtualBox:~/dir1$ cd ../dir2
agusnfr@agusnfr-VirtualBox:~/dir2$ ls ../dir1
f4
agusnfr@agusnfr-VirtualBox:~/dir2$
```

- Haga lo mismo que en el inciso anterior pero el archivo de destino, se debe llamar "f7".

```
agusnfr@agusnfr-VirtualBox:~/dir2$ cp f4 /home/agusnfr/f7
agusnfr@agusnfr-VirtualBox:~/dir2$ cd ..
agusnfr@agusnfr-VirtualBox:~$ ls
Descargas  Documentos  f2  Imágenes  Música  Público
dir1       Escritorio  f3  iso       Plantillas  snap
dir2       f1          f7  legajo    prueba    Videos
agusnfr@agusnfr-VirtualBox:~$
```

- Cree el directorio copia dentro del directorio usuario y copie en él, el contenido de "dir1".

```
agusnfr@agusnfr-VirtualBox:~$ mkdir copia
agusnfr@agusnfr-VirtualBox:~$ cp /home/agusnfr/dir1/* /home/agusnfr/copia
cp: -r not specified; omitting directory '/home/agusnfr/dir1/dir11'
agusnfr@agusnfr-VirtualBox:~$ cp /home/agusnfr/dir1/* /home/agusnfr/copia -r
agusnfr@agusnfr-VirtualBox:~$ cd copia
agusnfr@agusnfr-VirtualBox:~/copia$ ls
dir11  f4
agusnfr@agusnfr-VirtualBox:~/copia$
```

- Renombre el archivo "f1" por el nombre archivo y vea los permisos del mismo.

```
agusnfr@agusnfr-VirtualBox:~$ mv f1 archivo
agusnfr@agusnfr-VirtualBox:~$ ls
archivo  dir1      Escritorio  f7      legajo    prueba    Videos
copia    dir2      f2          Imágenes Música    Público
Descargas Documentos f3        iso      Plantillas snap
agusnfr@agusnfr-VirtualBox:~$
```

- Cambie los permisos del archivo llamado archivo de manera de reflejar lo siguiente:
  - Usuario: Permisos de lectura y escritura
  - Grupo: Permisos de ejecución
  - Otros: Todos los permisos



```

agusnfr@agusnfr-VirtualBox:~$ chmod 617 archivo
agusnfr@agusnfr-VirtualBox:~$ ls -l
total 60
-rw--xrwx 1 agusnfr agusnfr  0 sep  8 13:45 archivo
drwxrwxr-x 3 agusnfr agusnfr 4096 sep  8 14:01 copia
drwxr-xr-x 2 agusnfr agusnfr 4096 sep  7 21:41 Descargas
drwxrwxr-x 3 agusnfr agusnfr 4096 sep  8 13:50 dir1
drwxrwxr-x 2 agusnfr agusnfr 4096 sep  8 13:45 dir2
drwxr-xr-x 2 agusnfr agusnfr 4096 sep  7 19:31 Documentos
drwxr-xr-x 2 agusnfr agusnfr 4096 sep  8 11:45 Escritorio
-rw-rw-r-- 1 agusnfr agusnfr  0 sep  8 13:45 f2
-rw-rw-r-- 1 agusnfr agusnfr  0 sep  8 13:45 f3
-rw-rw-r-- 1 agusnfr agusnfr  0 sep  8 14:00 f7
drwxr-xr-x 2 agusnfr agusnfr 4096 sep  7 19:31 Imágenes
drwxrwxr-x 2 agusnfr agusnfr 4096 sep  8 12:48 iso
drwxrwxr-x 2 agusnfr agusnfr 4096 sep  8 12:30 legajo
drwxr-xr-x 2 agusnfr agusnfr 4096 sep  7 19:31 Música
drwxr-xr-x 2 agusnfr agusnfr 4096 sep  7 19:31 Plantillas
-rw-rw-r-- 1 agusnfr agusnfr  24 sep  8 13:12 prueba
drwxr-xr-x 2 agusnfr agusnfr 4096 sep  7 19:31 Público
drwx----- 5 agusnfr agusnfr 4096 sep  7 21:42 snap
drwxr-xr-x 2 agusnfr agusnfr 4096 sep  7 19:31 Videos
agusnfr@agusnfr-VirtualBox:~$

```

- Renombre los archivos "f3" "f4" de manera que se llamen "f3.exe" "f4.exe" respectivamente.

```

Descargas Documentos f3 iso Plantillas snap
agusnfr@agusnfr-VirtualBox:~$ mv f3 f3.exe
agusnfr@agusnfr-VirtualBox:~$ ls
archivo dir1 Escritorio f7 legajo prueba Videos
copia dir2 f2 Imágenes Música Público
Descargas Documentos f3.exe iso Plantillas snap

```

```

directorio
agusnfr@agusnfr-VirtualBox:~$ cd dir2
agusnfr@agusnfr-VirtualBox:~/dir2$ touch f4
agusnfr@agusnfr-VirtualBox:~/dir2$ mv f4 f4.exe
agusnfr@agusnfr-VirtualBox:~/dir2$ ls
f4.exe
agusnfr@agusnfr-VirtualBox:~/dir2$

```

- Utilizando un único comando cambie los permisos de los dos archivos renombrados en el inciso anterior, de manera de reflejar lo siguiente:
  - Usuario: Ningún permiso
  - Grupo: Permisos de escritura
  - Otros: Permisos de escritura y ejecución

Junte f4.exe y f3.exe en un mismo archivo e hice:

O puede ser `chmod 023 f3.exe f4.exe`

```

f3.exe f4.exe
agusnfr@agusnfr-VirtualBox:~/dir2$ sudo chmod -R 023 /home/agusnfr/dir2
agusnfr@agusnfr-VirtualBox:~/dir2$ sudo ls -l dir2
ls: no se puede acceder a 'dir2': No existe el archivo o el directorio
agusnfr@agusnfr-VirtualBox:~/dir2$ sudo ls -l
total 0
-----w--wx 1 agusnfr agusnfr 0 sep  8 14:22 f3.exe
-----w--wx 1 agusnfr agusnfr 0 sep  8 14:10 f4.exe
agusnfr@agusnfr-VirtualBox:~/dir2$

```

13. Indique qué comando/s es necesario para realizar cada una de las acciones de la siguiente secuencia de pasos (considerando su orden de aparición):

(a) Cree un directorio llamado logs en el directorio /tmp.

```
agusunfr@agusunfr-VirtualBox:/tmp$ mkdir /tmp/logs
agusunfr@agusunfr-VirtualBox:/tmp$ ls
logs
snap.snapd-desktop-integration
systemd-private-1b6abc43bf9b42c592b94d363cb3d6af-color.service-BEktRC
systemd-private-1b6abc43bf9b42c592b94d363cb3d6af-ModemManager.service-BXivEP
```

(b) Copie todo el contenido del directorio /var/log en el directorio creado en el punto anterior.

`cp /var/log/* /tmp/logs -R`

```
cp: -f not specified; omitting directory 'unattended-upgrades'
agusunfr@agusunfr-VirtualBox:/var/log$ cp * /tmp/logs -r
cp: no se puede abrir 'boot.log' para lectura: Permiso denegado
cp: no se puede abrir 'btm' para lectura: Permiso denegado
cp: no se puede acceder a 'gdm3': Permiso denegado
cp: no se puede acceder a 'private': Permiso denegado
cp: no se puede acceder a 'speech-dispatcher': Permiso denegado
agusunfr@agusunfr-VirtualBox:/var/log$
```

(c) Empaque el directorio creado en 1, el archivo resultante se debe llamar "misLogs.tar".

`tar -cvf misLogs.tar /tmp/logs`

(d) Empaque y comprima el directorio creado en 1, el archivo resultante se debe llamar "misLogs.tar.gz".

`tar -cvzf misLogs.tar.gz /tmp/logs`

(e) Copie los archivos creados en 3 y 4 al directorio de trabajo de su usuario.

`cp /tmp/misLogs.tar $HOME`

`cp /tmp/misLogs.tar.gz $HOME`

(f) Elimine el directorio creado en 1, logs.

`rmdir /tmp/logs`

(g) Desempaque los archivos creados en 3 y 4 en 2 directorios diferentes.

`tar -xvf misLogs.tar -C /home/agusunfr/dir1`

`tar -xvzf misLogs.tar.gz -C /home/agusunfr/dir2`