Practica 2

Siguiente
Anterior

El objetivo de esta práctica es que el alumno comprenda los aspectos principales acerca de la estructura del sistema Operativo GNU/Linux en lo que respecta a procesos, usuarios, filesystems, permisos, etc

1) Editor de textos:

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

- **Vim:** Es un editor de texto que rompe las bolas pero aprendes
- **GNU Emacs:** La misma basura que vim pero con calculadora y administrador de archivos
- **mcedit:** Te permite navegar entre los ficheros con una interfaz.

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

Los **comandos cat, more y less** permiten mostrar el contenido de ficheros de texto desde la línea de comandos en sistemas **Unix**.

En su lugar los **editores de texto**, justamente nos dejan **editar texto** además de poder **visualizarlo**.

- `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.
- `more` permite visualizar por pantalla el contenido de un fichero de texto, con la diferencia con el anterior de que `more` página 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.
- `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.

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

- Comandos basicos

---

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.

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

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

- Su función es cargar todos los subprocesos necesarios para el correcto funcionamiento del SO
- El proceso init posee el PID 1 y se encuentra en **/sbin/init**
- En SysV se lo configura a traves del archivo **/etc/inittab**
- No tiene padre y es el padre de todos los procesos (pstree)
- Es el encargado de montar los filesystems y de hacer disponible los dem´as dispositivos

---

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

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.

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

- Es el modo en que arranca Linux (3 en Redhat, 2 en Debian)
- El proceso de arranque lo dividimos en niveles
- Cada uno es responsable de levantar (iniciar) o bajar (parar) una serie de servicios
- Un nivel de ejecución es básicamente una configuración de programas y servicios que se ejecutarán orientados a un determinado funcionamiento.

Explicación más detallada

- `Paso 1)` 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**).
- `Paso 2)` 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.
- `Paso 3)` 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.
- `Paso 4)` 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
- `Paso 5)` Dentro del directorio encontraremos una serie de **scripts** o 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.

`(e)` ¿A qué hace referencia cada nivel de ejecución según el estándar? Fuente

- `0` Indica halt o apagado de la máquina.
- `1` Indica monousuario.
- `2` Indica modo multiusuario sin soporte de red.
- `3` Indica modo multiusuario completo con soporte de red.
- `4` No usado, con esta opción el administrador puede personalizar el inicio para cargar algún servicio.
- `5` Indica multiusuario completo con inicio gráfico (X11)
- `6` Indica shutdown y reboot: Se apaga inmediatamente la máquina para reinicio.

Un administrador (root) puede editar el archivo **/etc/inittab** como mejor convenga al usuario, sin embargo también tiene el poder de establecerlo en 0 o en 6. Si se establece en 6, algo que hice como experimento en mi Mandriva, la próxima vez que la máquina se encienda, se leerá el modo 6, shutdown y reboot, y se hará exactamente eso.

¿Dónde se define qué Runlevel ejecutar al iniciar el sistema operativo?

- Se encuentran definidos en **/etc/inittab**
- 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:

```
[SjK]<orden><nombreScript>
```

- `S)` lanza el script con el argument start
- `K)` lanza el script con el argument stop

¿Todas las distribuciones respetan estos estándares?

No todas las distribuciones respetan los estándares.

`(f)` Archivo /etc/inittab. ¿Cuál es su finalidad?

Es el archivo de configuración de init, que decide el modo de ejecución en el que va a entrar.

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.

¿Qué tipo de información se almacena en el? ¿Cuál es la estructura de la información que en él se almacena?

`/etc/inittab`

`id:nivelesEjecucion: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

`(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?

Existen dos formas de modificar los runlevels:

- **a) Cambiar de runlevel en ejecución:** Existe una utilidad para línea de comandos que permite cambiar de un nivel de ejecución a otro. Esta es la herramienta init. Para cambiar de nivel de ejecución sólo hay que ejecutar init seguido del número del runlevel.
  Por ejemplo
- `init 0` Cambia al runlevel 0 (se apaga el sistema, equivalente al comando halt).
- `init 2` Cambia al runlevel 2.
- `init 6` Cambia al runlevel 6 (reinicia el sistema, equivalente al comando reboot).

También `telinit`, nos permite cambiar de nivel de ejecución, sólo tenemos que indicar el número. Por ejemplo, necesitamos hacer una tarea crítica en root; sin usuarios trabajando, podemos hacer un `tellinit 1` (también puede usarse S) para pasar a runlevel monousuario, y después de la tarea un tellinit 3 para volver a multiusuario

---

`b) Modificar el runlevel por defecto`

Por defecto, el sistema suele arrancar en el nivel de ejecución 5 (modo gráfico). Si se quisiera modificar este comportamiento, habría que editar el fichero `/etc/inittab.`

Más concretamente, habría que modificar en el fichero /etc/inittab la línea donde el número 5 indica que el nivel de ejecución por defecto es el 5

No es permanente. En el caso de que el runlevel se cambie durante la sección de bash abierta y luego se apague la máquina, cuando se vuelva a prender la maquina se volverá a restablecer al modo que tenga el sistema configurado (por defecto).

En el caso de que se quiera cambiar el modo de arranque del runlevel de manera permanente se tendrá que configurar para que eso suceda.

```
ls /etc/rc0.d
sudo runlevel
```

```
sudo telinit 2
```

`(h)` Scripts RC. ¿Cuál es su finalidad?

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).

¿Dónde se almacenan?

Todos estos servicios se encuentran en **/etc/init.d/**

Sin embargo, no todos los servicios se cargan en todos los runlevels. **¿Cómo sabe el RC que servicios tiene que cargar?** 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/

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.

`Orden para llamarlos:`

Los nombres en estos directorios tienen una sintaxis bastante concreta. Empiezan por una 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 indica el orden en el que se arrancará el servicio.

- `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.

`(i)` ¿Qué es insserv?

El comando **insserv** se usa para controlar el orden de inicio y detención de los servicios que se encuentran en un sistema Linux.

¿Para qué se utiliza?

Se utiliza para administrar el orden de los enlaces simbólicos del **`/etc/rcX.d`**, resolviendo las dependencias de forma automática

- Utiliza cabeceras en los scripts del **`/etc/init.d`** que permiten especificar la relación con otros scripts rc -> LSBInit (Linux Standard Based Init)
- Es utilizado por update-rc.d para instalar / remover los links simbólicos

¿Qué ventajas provee respecto de un arranque tradicional?

Mejora la performance del arranque en sistemas multiprocesadores.

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

Upstart fue el primer reemplazo propuesto para SystemV (Ubuntu, Fedora, Debian, etc.).

- Permite la ejecución de trabajos en forma asincrónica a través de eventos (event-based) como principal diferencia con sysVinit que es estrictamente sincrónico (dependencybased).
- Estos trabajos se denominan **Jobs**.

- El principal objetivo de un job es definir servicios o tareas a ser ejecutadas por init
- Son scripts de texto plano que definen las acciones/tareas (unidad de trabajo) a ejecutar ante determinados eventos.
- Cada job es definido en el **/etc/init (.conf).**
- Suelen ser de dos tipos:
  - **Task**: ejecución finita (task) -> not respawning -> exit 0 o uso de stop.
  - **Service**: ejecución indeterminada respawning
- Los jobs son ejecutados ante eventos (arranque del equipo, inserción de un dispositivo USB,etc)
  - Es posible crear eventos pero existen algunos de manera estándar.
  - Definido por **start on y stop on.**
- Es compatible con SystemV ! **/etc/init/rc-sysinit.conf**, runlevels, scripts en /etc/init.d, objetivo start y stop.
- Cada job posee un objetivo (goal start/stop) y un estado (state).
  - En base a ellos se ejecuta un proceso específico.
  - Al inicio, init emite el evento startup.
- Un job puede tener uno o varias tareas ejecutables como parte de su ciclo de vida y siempre debe existir la tarea principal
- Las tareas de un job se definen mediante exec o script ... end script
- A través de **initctl** podemos administrar los jobs del demonio de Upstart:
- `start <job>`: cambia el objetivo a start del job especificado
- `stop <job>`: cambia el objetivo a stop del job especificado
- `emit <event>`: event es emitido causando que otros Jobs cambien a objetivo start o stop
- No más **/etc/inittab**

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

Upstart se creó como reemplazo del modelo SysVinit. A diferencia de SysVinit, que se creó para operar en un entorno estático Upstart se creó para operar en un entorno flexible.

Upstart proporciona beneficios principales sobre el SysVinit. Estos beneficios son: event-based (principal diferencia con SysVinit que es estrictamente sincrónico - dependecy-based -) es la ejecucion de servicios en forma asincrónica y otro beneficio es el reinicio automático de servicios que dejan de responder de la manera inesperada para el sistema.

Upstart en lugar de usar runlevels, usa jobs que cada uno de ellos posee un objetivo (start/stop y un estado state). Cuando ocurre una interrupción, upstart detecta ese interrupción y realiza los cambios necesarios.

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

- Los `jobs` reemplazan a los scripts de SystemV en Upstart.
- Cada job es definido en el `/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 Servise
description "MySQL Server " {Descripcion}
autor "info autor" {Autor}
start on ( net – device – up {Iniciar base de datos}
        and local –filesystems   {}
        and runlevel [2345])
stop on runlevel [016]
[...]
exec / usr / sbin /mysqld
[...]
```

Este es un archivo de configuración Upstart para el servicio de base de datos MySQL. Cada línea se explica a continuación:

- MySQL Service: un comentario que describe el servicio que se va a iniciar.
- description "MySQL Server": una descripción del servicio, que aparecerá en los registros del sistema. {Descripcion}
- author "info autor": información sobre el autor del archivo de configuración. {Autor}
- start on (net-device-up and local-filesystems and runlevel [2345]): indica que el servicio debe iniciarse cuando la red y el sistema de archivos locales estén disponibles y el sistema se esté ejecutando en el nivel de ejecución 2, 3, 4 o 5. {Iniciar base de datos}
- stop on runlevel [016]: indica que el servicio debe detenerse cuando el sistema se esté ejecutando en el nivel de ejecución 0, 1 o 6.
- exec `/usr/sbin/mysqld` : la línea que indica al sistema qué comando ejecutar para iniciar el servicio MySQL. El comando `/usr/sbin/mysqld` es el comando para iniciar el servidor de base de datos MySQL.

`(n)` ¿Qué es sytemd?

- Es un sistema que centraliza la administración de demonios y librerias del sistema.
- Puede ser controlado por `systemctl`
- Compatible con SysV -> si es llamado como `init`
- El demonio systemd reemplaza al proceso init -> este pasa a terner PID 1
- Los runlevels son reemplazados por `targets`
- Al igual que con Upstart el archivo `/etc/inittab` no existe más.

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

Las unidades de trabajo son denominadas units de tipo:

Service: controla un servicio particular (.service)

- **`Socket`** encapsula IPC, un sockect del sistema o file system FIFO (.socket) -> sockect-based activation.
- **`Target`** agrupa units o establece puntos de sincronización durante el booteo (.target)
- **`dependencia`** de unidades
- **`Snapshot`** almacena el estado de un conjunto de unidades que puede ser establecido más tarde (.snapshot) etc.

Las **units** pueden tener dos estados -> **active** o **inactive**

`(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.

---

3) Usuarios

`(a)` ¿Qué archivos son utilizados en un sistema GNU/Linux para guardar la información de los usuarios?

En un sistema GNU/Linux, la información de los usuarios se almacena principalmente en los siguientes archivos:

- `/etc/passwd` Este archivo contiene información básica de los usuarios, como sus nombres de usuario, identificación de usuario (UID), identificación de grupo (GID), nombre completo, ruta del directorio de inicio y shell predeterminada.
- `/etc/shadow` Este archivo contiene información confidencial de los usuarios, como sus contraseñas encriptadas, tiempo de última modificación de la contraseña, tiempo de expiración, cuenta bloqueada, etc.
- `/etc/group` Este archivo contiene información de los grupos de usuarios, como el nombre del grupo, identificación de grupo (GID) y una lista de nombres de usuario que pertenecen a ese grupo.
- `/etc/gshadow` Este archivo contiene información confidencial de los grupos de usuarios, como sus contraseñas encriptadas, tiempo de última modificación de la contraseña, tiempo de expiración, cuenta bloqueada, etc.

Es importante destacar que estos archivos son de lectura y escritura solo para el superusuario (root) y que modificarlos sin conocimiento puede comprometer la seguridad del sistema.

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

Los sistemas operativos Linux y Unix utilizan el UID (User ID o ID de usuario) para identificar al usuario particular. El GID (Group ID o ID de grupo) se utiliza para identificar a un grupo. Supongo que no podrian existir dos iguales ya que no los podrias distinguir.

Puede haber un caso que seria el root en el que podemos tener varios usuarios root con el ID 0

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

En sistemas operativos del tipo Unix, el superusuario o root es el nombre convencional de la cuenta de usuario que posee todos los derechos en todos los modos (monousuario o multiusuario). Normalmente es la cuenta de 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.

- `sudo adduser iso2022` creo un usuario y en home le agrego /home/ (contra = nombre para pruebas)
- `sudo gropadd catedra` creo un grupo
- `sudo usermod -a -G catedra iso2022`
- `id -nG iso2022` menciona los grupos a los que pertenece mi usuario
- `sudo login iso2022` entro como el usuario
- `cd ..` para ir a la home personal y crear un archivo (creo)
- `sudo userdel iso2022` lo elimina pero aun tenemos todos los archivos creados por este

---

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

- `useradd nombre ó adduser nombre` Crea un nuevo usuario
- `usermod nombre` nos permite modificar todos los parámetros de la cuenta de un usuario creado con anterioridad.
- `userdel nombre` Elimina un usuario
- `su` entrar al super usuario (tenes los permisos de TODO)
- `groupadd nombre` te deja crear un grupo
- `who` Verifiqua los usuarios conectado al sistema
- `groupdel nombre` elimina un grupo
- `passwd` de deja cambiar la constraseña del usuario actual

---

4) FileSystem:

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

Este mecanismo permite que archivos y directorios "pertenezcan" a un usuario en particular. Por ejemplo, como diego creó archivos en su directorio "home", diego es el propietario de esos archivos y tiene acceso total a ellos.

Unix también permite que los archivos sean compartidos entre usuarios y grupos de usuarios. Si diego lo desea, podría restringir el acceso a sus archivos de forma que ningún otro usuario pueda acceder a ellos.

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.

- `lectura` permite a un usuario leer el contenido del archivo o en el caso de un directorio, listar el contenido del mismo (usando ls).
- `escritura` permite a un usuario escribir y modificar el archivo (inclusive, eliminarlo). Para directorios, el permiso de escritura permite crear nuevos archivos o borrar archivos ya existentes en el mismo.
- `ejecución` permite a un usuario ejecutar el archivo si es un programa. Para directorios, el permiso de ejecución permite al usuario ingresar al mismo (por ejemplo, con el comando cd).
- `Interpretando los permisos de archivos` Veamos un ejemplo del uso de permisos de archivos. Usando el comando ls con la opción `-l` se mostrara un listado `largo` de los archivos, el cual incluye los permisos.

```
fabrizio@debian: /$ ls -l
-rwxr-xr-- 1 fabrizio users 505 May 5 19:05 prueba.exe
```

El primer campo representa los permisos del archivo. El tercer campo es el propietario del mismo (fabrizio), el cuarto es el grupo al cual pertenece el archivo (users) y el último campo es el nombre del archivo (prueba.exe).

La cadena `-rwxr-xr--` nos informa, por orden, los permisos para el propietario, los usuarios del grupo y el resto de los usuarios.

El primer carácter de la cadena de permisos `("-")` representa el tipo de archivo. El `"-"` significa que es un archivo regular, `"d"` indicaría que se trata de un directorio. Los siguientes tres caracteres `("rwx")` representan los permisos para el propietario del archivo, fabrizio. Éste tiene permisos para leer `(r)`, escribir `(w)` y ejecutar `(x)` el archivo prueba.exe.

Los siguientes tres caracteres, `"r-x"`, representan los permisos para los miembros del grupo al que pertenece el archivo (en este caso, users). Como sólo aparece `"r-x"` cualquier usuario que pertenezca al grupo users puede leer este archivo, y ejecutarlo, pero no modificarlo.

Los últimos tres caracteres, `"r--"`, representan los permisos para cualquier otro usuario del sistema (que no sea fabrizio ni pertenezca al grupo users). Como sólo está presente la `"r"`, los demás usuarios pueden leer el archivo, pero no escribir en él o ejecutarlo

Aquí tenemos otros ejemplos de permisos de grupo

- `-rw-------`
- El propietario del archivo puede leer y escribir. Nadie más puede acceder al archivo.
- `rwxrwxrwx`
- Todos los usuarios pueden leer, escribir y ejecutar el archivo.
- `drwxr-xr-x`
- El propietario del directorio puede leer, escribir y entrar al mismo. Los usuarios pertenecientes al grupo del directorio y todos los demás usuarios pueden leer e ingresar al directorio.

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

- **chmod:** nos permite gestionar permisos
- **chown:** permite cambiar el propietario de un archivo o directorio en sistemas
- **chgrp:** nos permite cambiar el grupo al que pertenece un archivo

`(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?

Existen 3 tipos de permisos y se basan en una notacion octal para referenciar a cada uno:

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

Se aplican sobre los usuarios:

- Usuarios: permisos del dueño -> `U`
- Usuarios: permisos del grupo -> `G`
- Usuarios: permisos de otro usuario -> `O`

La notación octal se refiere a ver estos valores en 3 bits , 010 equivale a 2 , por lo tanto si un archivo tendrá solo permisos de escritura, figura de esa manera. Si los valores fueran 110= serian permisos de lectura y escritura, y si fueran 111 el archivo tendría todos los permisos (lectura escritura y ejecución).

- En general cuando se le da un permiso a un archivo requiere 3 números (ejemplo `chmod 755`) cada uno de esos números representa los permisos para diferentes cosas.
- El primer número es para indicar los permisos para el dueño del archivo (`U`).
- El segundo número es para indicar los permisos para los usuarios del grupo de un archivo (`G`)
- El tercer número es para indicar los permisos para todo el resto de los usuarios (`O`).

Por lo tanto, en el ejemplo `chmod 755` , indica que el `dueño (U)` tiene permiso para lectura, escritura y ejecución (4 + 2 + 1 = 7). Los `usuarios del grupo (G)` y para el `resto de los usuarios (O)` tienen permiso para lectura y ejecución (4 + 1= 5). Ver tabla.

`(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.

Existe la posibilidad, si es que el usuario puede utilizar el comando su, o sudo. Sino, sin poseer los permisos necesarios no puede acceder al archivo. Sólo root.

`(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 completa a ese archivo o carpeta desde el directorio / del sistema de archivos. ejemplo `/home/your_username/my_script`
- `Relative path name` Rastrea la ruta desde el directorio actual a través de su padre o sus subdirectorios y archivos. ..\Documents

---

`(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 el comando pwd podemos saber el directorio actual.
- Con `cd` o `cd ~` vamos al directorio personal.
- Se puede usar este mismo comando para acceder a directorios a partir de la ruta en donde estemos parados

Se podría acceder a diferentes directorios gracias la ubicación relativa o atajos ya prestablecidos como `cd ..` para volver al directorio anterior sin necesidad de poner ningún atajo

---

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

- `cd` Nos permite meternos en un directorio interno
- `umount` permite eliminar un sistema de archivos remoto que esté montando en la actualidad (no usar xd)
- `mkdir` Cree una carpeta
- `du` Para ver el tamaño de ficheros y carpetas
- `rmdir` Borrar directorios
- `df` Informa la cantidad de espacio libre en disco
- `mount` Se utiliza para montar dispositivos y particiones para su uso por el sistema operativo (se instala con **sudo apt install nfs-common**)
- `ln` crear un enlace simbólico al fichero o directorio (como un acceso directo)
- `ls` Lista el contenido del directorio actual
- `pwd` Visualiza la ruta donde estoy situado
- `cp` sirve para copiar archivos y directorios dentro del sistema de archivos
- `mv` se utiliza para mover o renombrar los archivos y directorios

---

5) Procesos

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

Un proceso es un programa en ejecución. Para nosotros serán sinónimos de tarea, job y proceso.

- Es dinámico
- Tiene program counter.
- Su ciclo de vida comprende desde que se lo `dispara` hasta que termina.

La sigla `PID` hace referencia al `ID` del Proceso y la sigla `PPID` hace referencia al `ID` del Proceso Padre del proceso.

Todos los procesos tienen estos atributos, además de estos (que son los más importantes pero no todos): Usuario (`UID`), Grupo (`GID`), Prioridad, etc. Con ps –ejH

PUEDO VER EL PPID.

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

- `pstree` que nos muestra los procesos en una estructura de árbol top El comando top te permite ver las tareas del sistema que se ejecutan en tiempo real.
- `top` El comando top te permite ver las tareas del sistema que se ejecutan en tiempo real.Proporciona un buen resumen de tu sistema para verificar rápidamente si algo se destaca que pueda estar causando problemas con tu sitio web o servidor
- `ps` muestra por pantalla un listado de los procesos que están ejecutándose en el sistema.

Las opciones (parámetros) más importantes y utilizadas de este comando son:

- `a` para mostrar los procesos de todos los teminales.
- `u` para mostrar el usuario al que pertenece el proceso y la hora de inicio.
- `x` para mostrar procesos que no estan controlados por ningún terminal.

Suelen usarse combiandas para tener una visión global de los procesos que están en ejecutan.

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

- `Proceso ejecutándose en Background` significa que el proceso continúa la ejecución mientras que el shell se libera para otras actividades. `Proceso en segundo plano`.
- `Proceso ejecutándose en Foreground` Es exactamente lo contrario, quiere decir que no se admitirá ningún otro comando hasta que se complete el proceso. `Proceso en primer plano`.

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

Para ejecutar un proceso en background en Linux, se puede agregar el símbolo "&" al final del comando en la línea de comandos. Por ejemplo, para ejecutar un proceso llamado "mi_proceso" en background, se puede escribir:

```
mi_proceso &
```

Esto permitirá que el proceso se ejecute en segundo plano mientras se sigue usando la terminal para ejecutar otros comandos.

Para pasar un proceso de background a foreground y viceversa, se pueden usar los siguientes comandos:

- `fg` mueve un proceso en background a foreground. Se debe proporcionar el número de trabajo (job number) del proceso como argumento. El número de trabajo se puede obtener con el comando "jobs". Por ejemplo:

  ```
  fg %1
  ```

  Esto mueve el proceso número 1 en background al foreground.
- `bg` mueve un proceso en foreground a background. También se debe proporcionar el número de trabajo del proceso como argumento. Por ejemplo:

  ```
  bg %1
  ```

  Esto mueve el proceso número 1 en foreground a background.

También se puede suspender un proceso en foreground con la combinación de teclas `Ctrl + Z`. Esto detendrá el proceso y lo moverá a background. Para reanudar el proceso en background, se puede usar el comando `bg`

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

El `|` nos permite comunicar dos procesos por medio de un pipe o tubería 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 comanda `more`.
- Se pueden anidar tantos pipes como se deseen

¿Cómo haríamos si quisiéramos contar la cantidad de usuarios del sistema que en su nombre de usuario aparece una letra “a"?

```
cat /etc/passwd | cut -d: -f1 | grep a | wc –l
```

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

En 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:

- `0` 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. Veamos unos ejemplos

- `ls -l > fichero` Guarda la salida de ls -l en fichero. Si no existe lo crea, y si existe lo sobreescribe.
- `ls -l >> fichero` Añade la salida del comando a fichero. Si no existe lo crea, y si existe, lo añade al final.
- `ls -l 2 > fichero` Si hay algún error, lo guarda en fichero (podría salir un error si no tuviéramos permiso de lectura en el directorio).

Las **redirecciones** consisten en trasladar información de un tipo a otro

Hay 2 tipos de redirecciones

- Al utilizar redirecciones mediante > (destructiva):
  - Si el archivo de destino no existe, se lo crea
  - Si el archivo existe, se lo trunca y se escribe el nuevo contenido
- Al utilizar redirecciones mediante >> (no destructiva):
  - Si el archivo de destino no existe, se lo crea
  - Si el archivo existe, se agrega la información al final

EJEMPLOS

> Redirecciona **stdout** hacía un archivo. Lo crea si no existe, si existe lo sobreescribe.

```
ls -l > lista.txt
```

> > (La salida del comando se envía a un archivo en vez de la terminal.)

Redirecciona **stdout** hacía un archivo. Lo crea si no existe, si existe concatena la salida al final de este.

```
ps -ef >> processos.txt
```

(Concatena al archivo procesos.txt la salida del comando.)

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 pasaría 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.

El comando `kill` en Linux (ubicado en / bin / kill), es un comando incorporado que se usa para terminar los procesos manualmente. El comando kill envía una señal a un proceso que termina el proceso. Si el usuario no especifica ninguna señal que se enviará junto con el comando kill, se envía la señal TERM predeterminada que finaliza el proceso.

- `kill -l` Para mostrar todas las señales disponibles, puede usar la siguiente opción de comando.
- `kill pid` Para mostrar cómo usar un PID con el comando kill.
- `kill -s` Para mostrar cómo enviar señales a los procesos.
- `kill -L` este comando se usa para listar las señales disponibles en un formato de tabla.

`(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.
- **kill:** Usa el PID para matar el proceso. Permite interactuar con cualquier proceso mandando señales. Kill `<pid>` termina un proceso y Kill -9 `<pid>` fuerza a terminar un proceso en caso de que la anterior opción falle.
- **pstree:** muestra un árbol de procesos.
- **killall:** nos permite matar un proceso escribiendo su nombre
- **top:** Sirve para ver los procesos de ejecución del sistema (y más cosas) en tiempo real
- **nice:** Ejecuta un comando con una prioridad determinada, o modifica la prioridad a de un proceso. `nice -10 named` Esto bajaría la prioridad de named en 10 unidades(Si estaba en -10, pasará 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?

Los archivos `TAR` no son archivos comprimidos sino empaquetados. `TAR` es un empaquetador, es algo más parecido a un compresor como `arj` ó `zip` pero sin compresión. Su función es la de incluir todos los ficheros juntos en el mismo archivo, conservando las estructuras de directorios y permisos de los mismos.

- `crear` un archivo .tar

```
tar –cvf mi_archivo.tar /directorio/a/empaquetar
```

- `extraer` el contenido de un archivo .tar ya creado

```
tar –xvf mi_archivo.tar
```

- `actualizar` el contenido de un archivo .tar ya existente

```
tar –uvf mi_archivo.tar
```

- `agregar` un directorio a un archivo .tar ya existente

```
tar –rvf archivo. tar  directorio/a/agregar
```

- `empaquetar` y comprimir el archivo .tar

```
tar –cvzf archivo.tgz /directorio/a/empaquetar/y/comprimir
```

- `desempaquetar` el archivo .tgz

```
tar –xvzf archivo.tgz
```

- Ojo : Observa la extensión ".tgz" para el tar comprimido. Aunque también podemos ponerle como extensión ".tar.gz"

`(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?

Claro, para sumar el tamaño de cuatro archivos en Linux y crear un archivo empaquetado que los contenga, puede seguir los siguientes pasos:

- Abrir una terminal y navegar hasta el directorio que contiene los cuatro archivos usando el comando `cd`.
- Para sumar el tamaño de los cuatro archivos, use el comando

```
du -sh file1 file2 file3 file4
```

  Esto le dará la suma total de los tamaños de los cuatro archivos.
- Para crear un archivo empaquetado que contenga los cuatro archivos, puede utilizar el comando

```
tar -cvzf archivo_empaquetado.tar.gz file1 file2 file3 file4
```

  Esto creará un archivo empaquetado llamado `archivo_empaquetado.tar.gz` que contiene los cuatro archivos.
- Para comparar los tamaños de los cuatro archivos individuales y el archivo empaquetado, use el comando `ls -lh`. Esto mostrará el tamaño de cada archivo en una lista. Compare el tamaño total de los cuatro archivos individuales con el tamaño del archivo empaquetado para notar la diferencia en el tamaño.

Es importante tener en cuenta que el comando `tar` utiliza la compresión para reducir el tamaño del archivo empaquetado, por lo que es posible que el tamaño del archivo empaquetado sea menor que la suma total de los tamaños de los cuatro archivos individuales.

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

Teniendo en cuenta que tenemos 4 archivos en el directorio Descargas/ejercicio6 y nos situamos en Descargas ejecutamos lo siguiente

```
tar cvfz archivo.tar.gz ejercicio6
```

Esto nos dejaria un archivo empaquetado de archivos comprimidos en un solo archivo archivo.tar.gz y para acceder a dicha informacion haremos

```
tar xvfz archivo.tar.gz
```

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

Si los archivos están en el mismo directorio utilizamos el comando visto anteriormente

---

(e) Investigue la funcionalidad de los siguientes comandos:

- `tar` Empaqueta/desempaqueta varios archivos en uno solo, puede realizar compresión sin perdida
- `grep` El comando grep nos permite buscar cadenas de texto y palabras dentro de un fichero de texto o de la entrada estándar de la terminal. Una vez encontrado el contenido que estamos buscando: grep mostrará en pantalla la totalidad de la línea/s que contiene/n la cadena de texto o palabra que estamos buscando
- `gzip` Comprime solo archivos utilizando la extensión .gz que se utiliza para truncar el tamaño de un archivo.
- `zgrep` Se usa para buscar expresiones de un archivo dado, incluso si está comprimido
- `wc` Cuenta los caracteres, palabras y líneas del archivo de texto.

---

7) Ejercicio

Enunciado: 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

```
l s –l > prueba {No se puede acceder a pruebas pq no existe el fichero}
ps > PRUEBA
chmod 710 prueba
chown root : root PRUEBA
chmod 777 PRUEBA
chmod 700 / etc / passwd
passwd root
rm PRUEBA
man / etc / shadow
find / –name * .conf
usermod root –d /home/ newroot –L
cd / root
rm *
cd / etc
cp * /home –R
shutdown
```

- `ls -l > prueba` Genera un archivo de nombre prueba que contiene un listado detallado con los contenidos del directorio home del usuario. Se redirige la salida estándar de ls mediante el carácter > hacia el archivo prueba.
- `ps > PRUEBA` Genera un archivo de nombre PRUEBA que contiene un listado de los procesos en ejecución en el directorio home del usuario. AL igual que en el ejemplo anterior, se redirige la salida estándar mediante >.
- `chmod 710 prueba` Cambia los permisos del archivo prueba a 710 para UGO (usuario, Grupo, Otros).
- `chown root:root PRUEBA` Se intenta cambiar el propietario del archivo prueba pero la operación no está permitida.
- `chmod 777 PRUEBA` Cambia los permisos del archivo PRUEBA a 777. Es decir, todos los usuarios pueden leer, escribir y ejecutar el archivo.
- `chmod 700 /etc/passwd` Intenta cambiar los permisos a 700, pero la operación no está permitida para un usuario que no es root, esto por el archivo que está intentando cambiar.

- `passwd root` passwd: No debe ver o cambiar la información de la contraseña para root.
- `rm PRUEBA` Se elimina el archivo PRUEBA.
- `man /etc/shadow` Permiso denegad, porque "man" no debe recibir una ruta, si hago "man shadow" si anda.
- `find / -name *.conf` Lista todos los archivos cuyo nombres terminan con .conf, empezando la búsqueda en el directorio raíz /.
- `usermod root –d /home/newroot –L`
- `cd /root` Se intenta acceder a la carpeta root, pero la operación falla porque el usuario no tiene los permisos.
- `rm *` Borra todos los archivos del directorio donde está posicionado el usuario.
- `cd /etc` Cambia el directorio a /etc, osea "se mueve" a /etc
- `cp * /home –R` Intenta copiar el contenido de /etc a home, pero el usuario no tiene los permisos necesarios para crear archivos en el directorio /home.
- `shutdown` Apaga el equipo

---

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

(a) Terminar el proceso con PID 23.

- Todos los procesos tienen un pid (id de proceso) para terminar un proceso se usa el comando:

```
kill -9 23 (kill menos 9 número de id del proceso).
```

- `ps –aux` Para ver los procesos del sistema que corren en el momento comando:

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

- `1)` Primero buscamos el PID de init con el comando `ps –aux`
- `2)` Ingresamos como superusuario: su – e ingresamos contraseña
- `3)` Utilizamos el comando kill –9 1
- `4)` Y vamos a ver que no va a suceder nada, porque el proceso init, no puede terminarse, así que ni responde al comando.

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

Para buscar un `archivo` se utiliza el comando:

```
find / -type f -name NombreDelArchivoABuscar
```

Búsqueda por tipo

Linux permite a los usuarios listar toda la información basada en sus tipos. Hay varios filtros que puedes usar:

- `d` directorio o carpeta
- `f` archivo normal
- `l` enlace simbólico
- `c` dispositivos de caracteres
- `b` dispositivos de bloque

Si deseamos buscar la palabra "conf" en todo el sistema se utiliza el comando:

```
find / -name NombreABuscar
```

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

```
ps > /home/user/procesos
```

- `cat Procesos` Para poder comprobar el contenido que tiene el directorio Procesos
- `rm –r Procesos` Para eliminar dicho directorio

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

- **Usuario:** Lectura, escritura, ejecución
- **Grupo:** Lectura, ejecución
- **Otros:** ejecución
- `mkdir xxxx` Primero creamos el directorio “xxxx”
- `ls –l` Para ver los permisos que tiene dicho directorio
- `chmod 751 xxxx` Para cambiar sus permisos donde Usuario tenga los permisos de lectura, escritura, ejecución, Grupo los permisos de lectura y ejecución y Otros el permiso
- `7` = 4(Lectura) + 2(Escritura) + 1(Ejecución) --> `Usuario`
- `5` = 4(Lectura) + 1(Ejecución) --> `Grupo`
- `1` = 1(Ejecución) --> `Otros`
- `ls -l` Para chequear si los permisos fueron modificados

Para lograr visualizar los permisos se lee de la siguiente forma:

```
drwxr-x--x 2 lucia lucia 4096 sep 30 19:15 xxxx
```

- No ver la primera d!
- Los primeros `rwx` (Lectura, Escritura, Ejecución) pertenecen a Usuario
- Luego rx (LecturaEjecución) pertenece a Grupo
- Y x (Ejecución) pertenece a Otros

(f) Cambiar los permisos del archivo /home//yyyy a:

- **Usuario:** Lectura, escritura.
- **Grupo:** Lectura, ejecución
- **Otros:** Ninguno
- `mkdir yyyy` Primero creamos el directorio `“yyyy”`
- `ls –l` Para ver los permisos que tiene dicho directorio
- `chmod 650 yyyy` Para cambiar sus permisos donde Usuario tenga los permisos de lectura, escritura, ejecución, Grupo los permisos de lectura y ejecución y Otros el permiso de ejecución.

(g) Borrar todos los archivos del directorio /tmp

```
cd /tmp
rm *
```

(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 encuentro en este momento, en caso de que el archivo exista no se debe eliminar su contenido anterior.

```
pwd >> /home/user/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.

```
sudo adduser il {crear}
passwd il {nueva contra}
```

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

se modificaron los archivos `/etc/passwd` y se creo el directorio personal del perfil en /home/nombrelegido

---

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

```
cd /tmp
mkdir cursada2017
```

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

En modo superusuario/root

```
cp /var/log/* /tmp/cursada2017
```

(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 nombreUsuario:nombreUsuario /tmp/cursada2017
```

(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 -R 745 /tmp/cursada2017
```

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

```
sudo login user
```

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

```
ps -p 544
```

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

```
ps aux | wc -l
```

(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.

```
sudo shutdown 1 'El sistema se va a apagar'
tmb sin apagar
wall "En un minuto apagaremos el sistema."
```

(m) Apague el sistema

```
sudo shutdown now
```

---

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 19508/3 {No se puede dado q confunde / con un desplazamiento de directorio}
```

(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".

```
cd legajo
vi LEAME
```

(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

```
chmod 017 LEAME
```

(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 cuál puede crear este archivo si ya existe un archivo llamado "LEAME.en este directorio?.

Se puede porque unix es case sensitive y distingue entre mayúsculas y minúsculas.

(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 / -name "[0-9]*"
```

(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".

```
find / -name "*.os" > /home/user/legajo/ejerciciof
```

---

11) Ejercicio

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 el directorio iso
- `cd . / iso; ps > f0` Me situo en iso y creo el archivo f0 con los procesos en ejecucion
- `ls > f1` Guardo en F1 la lista de archivos en el directorio iso
- `cd /` Me posiciono en la raiz
- `echo $HOME` muestra en pantalla la direccion del directorio personal del usuario
- `ls –l $> $HOME/iso/ls` No se puede acceder a $ no existe el fichero o directorio

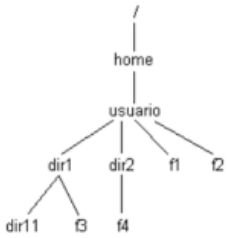
- `cd $HOME; mkdir f2` Nos situamos en el directorio personal y creamos el directorio f2
- `ls -ld f2` Se da un listado detallado de f2
- `chmod 341 f2` Se modifican los permisos de f2;Es,Ej para usuario,L para Grupo, y Ej para otros
- `touch dir` se crea el archivo dir
- `cd f2` accedemos al directoriio f2
- `cd ~/ iso` accedemos a la carpeta iso
- `pwd > f3` se guarda en el archivo f3 la direccion de la carpeta iso
- `ps | grep ' ps ' | wc -l >> ../f2/f3` se obtienen los procesos, se usa de entrada para grep el cual filtra todos los que posean ps,pra luego almacenar la cantidad de lineas obtenidas luegro de dicho filtro y añadirlas al archivo f3
- `chmod 700 .. / f2 ; cd ..` Se modifica el acceso de f2 dando al usuario todos los permisos y a los demas nada, volviendo despues al directorio personal
- `find . -name etc / passwd` Lanza un warning por mal uso del comando
- `find / -name etc / passwd` filtra todos los archivos de passwd por nombre en orden
- `mkdir ejercicio5` crea el directorio 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. (ARRIBA)

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

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



---

12) Ejercicio

Enunciado: 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:

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

```
mv f3 $HOME
```

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

```
cp f4 $HOME/dir11
```

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

```
cp f4 $HOME/dir11/f7
```

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

```
mkdir copia; cp -a dir11 copia
```

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

```
cd iso;mv f0 archivo; ls  -ld archivo
```

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

```
chmod 617 archivo
```

Renombre los archivos "f3 2 "f4"de manera que se llamen "f3.exe 2 "f4.exerespectivamente.

```
mv f3 f3.exe; cd $HOME/dir11 ; mov f4 f4.exe
```

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

```
chmod 023 f3.exe f4.exe
```

`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.

```
cd /tmp; mkdir logs
```

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

```
cp -a /var/log/. /tmp/logs
```

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

```
tar cvf misLogs.tar logs
```

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

```
tar cvfz misLogs.tar.gz logs
```

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

```
cp misLoggs.tar $HOME
cp misLogs.tar.gz $HOME
```

(f) Elimine el directorio creado en 1, logs

```
rm -r logs
```

(g) Desempaquete los archivos creados en 3 y 4 en do directorios diferentes.

```
tar xvf misLogs.tar -C 1
tar xvfz misLogs.tar.gz -C 2
```

Fecha

Con el comando `Date`