

# Introducción a los Sistemas Operativos

## Práctica 2

**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.**

- VI
- VIM
- NEOVIM
- NANO
- EMACS
- JOE

**(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 son comandos de visualización de archivos en la consola de Linux, mientras que los editores de texto por consola como Vi se utilizan para editar archivos de texto.

El comando cat se utiliza para mostrar el contenido de un archivo en la consola. El contenido del archivo se muestra en su totalidad, lo que puede ser problemático para archivos grandes.

El comando more se utiliza para mostrar el contenido de un archivo en la consola, pero a diferencia del comando cat, muestra el contenido del archivo página por página. El usuario puede navegar por el archivo utilizando las teclas de flecha o la barra espaciadora.

El comando less es similar al comando more, pero es más eficiente y rápido para archivos muy grandes. En lugar de cargar todo el contenido del archivo y mostrarlo según lo solicitado, carga solo lo que se necesita a medida que se navega por el archivo.

Por otro lado, los editores de texto por consola como Vim, Emacs, Nano y Joe permiten al usuario editar archivos de texto directamente desde la consola. Estos editores tienen características adicionales como resaltado de sintaxis, autocompletado, pantalla dividida en varios paneles, etc.

Hay dos modos de operar en vi: modo de inserción y modo de comando:

En el modo de comandos, podemos desplazarnos dentro de un archivo y efectuar operaciones de edición como buscar texto, eliminar texto, modificar texto, etc. Vi suele iniciarse en modo de comandos

En el modo insertar, podemos escribir texto nuevo en el punto de inserción de un archivo. Para volver al modo de comandos, presione la tecla esc.

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

Para usar vi, estos son los comandos esenciales para escribir, editar, borrar, copiar y pegar.

### **Editar y modificar**

- Para insertar texto antes del cursor: i
- Para insertar texto después del cursor: a
- Para insertar texto al principio de la línea: I
- Para insertar texto al final de la línea: A


### **Copiar y pegar**

- Para copiar la línea actual: yy
- Para copiar una palabra: yw
- Para copiar 7 líneas: y7y
- Para pegar después del cursor: p
- Para pegar antes del cursor: P

### **Borrar**

- Para borrar un carácter: x o X
- Para borrar la línea actual: dd

### **Abrir, guardar y salir**

- Para abrir un archivo: :e nombre\_de\_archivo
- Para guardar los cambios y salir: :x o :wq o  Mayús+ZZ
- Para salir: :q
- Para salir sin guardar (forzar la salida): :q!
- Para guardar con otro nombre de archivo: :w nombre\_de\_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)

El Scheduler es el programa dentro del sistema operativo que administra de manera eficiente el procesador, es parte del núcleo del sistema operativo. El Planificador de trabajos o Scheduler se encarga de elegir la tarea siguiente que hay que admitir en el sistema y el proceso siguiente que hay que ejecutar

**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?**

En sistemas tipo Unix, init (abreviatura de initialization) es el primer proceso en ejecución tras la carga del kernel y el que a su vez genera todos los demás procesos.

- Lo ejecuta el kernel
- Su función es cargar todos los subprocessos necesarios para el correcto funcionamiento del SO
- Es el encargado de montar los filesystems y de hacer disponible los demás dispositivos.
- El proceso init posee el PID 1 y se encuentra en /sbin/init
- Se ejecuta como demonio.
- En SysV se lo configura a través del archivo /etc/inittab
- No tiene padre y es el padre de todos los procesos (pstree)

PID = Process IDentifier

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

El comando **pstree** es un comando de Linux que muestra los procesos en ejecución como un árbol jerárquico. Es una forma más conveniente de mostrar la jerarquía de procesos y hace que la salida sea más atractiva visualmente. El nodo raíz del árbol es el proceso init o el proceso con el PID especificado.

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

Un RunLevels se puede definir como un número entero preestablecido de un solo dígito para definir el estado operativo de su sistema operativo basado en LINUX o UNIX. Cada nivel es responsable de levantar (iniciar) o bajar (parar) una serie de servicios de ejecución, es decir designa una configuración del sistema diferente y permite el acceso a diferentes combinaciones de procesos.

Cada sistema Linux tiene un RunLevel de arranque predeterminado (3 en Redhat, 2 en Debian).

**(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**

Se encuentran definidos en `/etc/inittab`

Los runlevels son una característica común en la mayoría de las distribuciones de Linux, pero no todas las distribuciones respetan los mismos estándares de runlevels . Por ejemplo, algunos sistemas operativos como Arch Linux y Gentoo no utilizan runlevels en absoluto, mientras que otros sistemas operativos como Debian y Ubuntu utilizan runlevels personalizados que difieren de los estándares de Unix.

Sin embargo, la mayoría de las distribuciones de Linux siguen los estándares de Unix para los runlevels

**(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?**

El archivo /etc/inittab es un archivo de configuración utilizado por el proceso init en sistemas operativos tipo Unix, como Linux. El archivo define los procesos que se ejecutan en el sistema durante el proceso de arranque y define los runlevels que se utilizan para iniciar los servicios del sistema.

La estructura interna es la siguiente:

- **Id:** Es un identificador único para la entrada
- **state:** Muestra los niveles de ejecución a los que se aplica esta entrada
- **action:** Identifica el modo en que el proceso que está especificado en el campo del proceso se ejecutará. Es decir, describe la información a realizar  
Los valores posibles incluyen:
  - sysinit
  - wait
  - initdefault
  - ctrlaltdel
  - boot
  - bootwait
  - wait
  - respawn
  - off, once, powerwait, etc.
- **process:** Define el comando o la secuencia de comandos para ejecutar.  
Dice 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?**

Para cambiar del runlevel 3 al runlevel 1, se puede utilizar el comando **init** seguido del número del runlevel.

Este cambio no es permanente y solo afecta al estado actual del sistema.

Puedo cambiarlo nuevamente de forma manual ingresando el comando, o en su defecto, el sistema volverá al runlevel predeterminado en el próximo arranque.

**(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.**

Cuando init entra en un nivel de ejecución, llama al script RC (de Run Control) con un argumento numérico que especifica el nivel de ejecución al que se va a ir. A continuación, RC inicia y detiene los servicios en el sistema según sea necesario para llevar el sistema a ese nivel de ejecución

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>

Donde S es de Start (inicia) y K de Kill (termina)

En cuanto al orden en que se llaman los scripts RC, los scripts se llaman en orden alfabético dentro del directorio /etc/rc.d o /etc/rcX.d (donde X es el número del nivel de ejecución). Los scripts que comienzan con una letra anterior en el alfabeto se llaman antes que los scripts que comienzan con una letra posterior. Además, los scripts que tienen un número más bajo después del primer carácter también se llaman antes que los scripts con un número más alto. Este orden garantiza que los servicios críticos se inicien antes que los servicios no críticos y que los servicios dependientes se inicien después de sus dependencias.

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

Insserv es un comando de Linux que 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)

Los scripts deben cumplir LSB init script:

- Proveer al menos 'start, stop, restart, force-reload and status'
- Retornar un código apropiado
- Declarar las dependencias

La ventaja de Insserv es que puede evitar problemas de dependencias y mejorar la performance.

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

Upstart utiliza un modelo basado en eventos que le permite responder a los eventos de forma asíncrona cuando estos son generados 1. 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.

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

Upstart tiene como objetivo reemplazar los daemons tradicionales de SystemV que gestionan las tareas a ejecutar en el arranque, la parada y puesta en marcha de servicios.

Permite la ejecución de trabajos (jobs) en forma asincrónica a través de eventos (event-based) como principal diferencia con sysVinit que es estrictamente sincrónico (dependency-based).

Es decir, SystemV utiliza un modelo secuencial de inicio, mientras que Upstart utiliza un modelo basado en eventos. En el modelo secuencial, los servicios se inician en un orden predefinido, mientras que en el modelo basado en eventos, los servicios se inician en respuesta a eventos específicos

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

Los **jobs** de Upstart reemplazan a los scripts RC de SystemV.

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

Los jobs pueden ser tareas o servicios. Las **task** son simples scripts que se ejecutan y terminan su trabajo en un breve lapso de tiempo. Los **services** son procesos que se quedan en ejecución en segundo plano (daemons) durante un tiempo indeterminado. Para que un job pueda ponerse en marcha o pararse, deben darse ciertas condiciones. Normalmente estas condiciones vienen dadas por la detección o no de ciertos eventos.

Los scripts de trabajo se almacenan en el directorio **/etc/init** en el sistema de archivos.

(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 autor"
start on (net-device-up
          and local-file-systems
          and runlevel[2 3 4 5])
stop on runlevel[0 1 6]

[...]

exec /usr/sbin/mysqld

[...]
```

---

**Starton():** Establece que el servicio debe iniciarse cuando se cumplan las condiciones dadas entre parentesis. En este caso, que el servicio de red este activo, que los archivos del sistema esten listos y que el sistema esté en runlevel 2, 3, 4 o 5.

**stoponrunlevel[ ] :** Establece que el trabajo debe detenerse en un runlevel. En este caso 0, 1 o 6

**exec ' ':** Establece algo a ejecutar. En este caso, el archivo que se encuentra en /usr/sbin/mysqld.

## **(n) ¿Qué es sytemd?**

Systemd es un sistema que interactúa con el Kernel y centraliza la administración de daemons y librerías del sistema.

Fue desarrollado para reemplazar el sistema de inicio (init) heredado de los sistemas operativos estilo UNIX System V.

Systemd es un sistema y administrador de servicios que se ejecuta como el primer proceso en el arranque del sistema (como PID 1) y mantiene los servicios del espacio de usuario.

Los runlevels son reemplazados por targets.

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

La activación de socket en systemd es un mecanismo que permite a los servicios ser iniciados bajo demanda, es decir, solo cuando se recibe una conexión en un socket específico. En lugar de mantener un servicio en ejecución continuamente, la activación de socket permite que el servicio se inicie solo cuando es necesario, lo que puede ahorrar recursos del sistema.

Cuando se recibe una conexión en el socket, systemd inicia el servicio correspondiente y le pasa el socket.

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

El concepto de cgroup hace referencia a un mecanismo en el kernel de Linux que limita, contabiliza y aísla el uso de recursos (CPU, memoria, E/S de disco, etc.) de un conjunto de procesos.

Cgroups (control groups) es una característica del kernel de Linux que permite limitar y distribuir los recursos del sistema entre los procesos . Los grupos de control pueden ser jerárquicos, lo que significa que cada grupo hereda límites de su grupo padre. Agrupa conjunto de procesos relacionados (por ejemplo, un servidor web Apache con sus dependientes)

Cgroups se compone principalmente de dos partes: el núcleo y los controladores. El núcleo es responsable de organizar jerárquicamente los procesos, mientras que los controladores son responsables de limitar y contabilizar 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 en varios archivos y directorios.

**/home:** Este directorio contiene las carpetas personales de los usuarios. Cada usuario tiene su propio directorio en /home, donde puede almacenar archivos personales y configurar su entorno de trabajo.

**/etc:** Este directorio contiene archivos de configuración del sistema. Aquí se encuentran los archivos que definen la configuración de los usuarios, como el archivo /etc/passwd, que contiene la lista de todos los usuarios del sistema.

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

UID = User Identifier

GID = Group Identifier)

Grupo se refiere a un conjunto de usuarios. Cada usuario puede pertenecer a uno o más grupos. Los grupos se utilizan para gestionar los permisos y los derechos de acceso a los recursos del sistema. Por ejemplo, puedes tener un grupo de usuarios que tienen permisos para acceder a ciertos archivos o directorios, mientras que otros usuarios no pueden

En un sistema Linux, cada usuario se identifica por un número único conocido como Identificador de Usuario (UID). Este número debe ser único para cada usuario en el sistema. Por lo tanto, no deberían poder coexistir dos UID iguales en el mismo sistema, ya que cada UID está asociado con un usuario específico. Esto es importante para la gestión de permisos y el control de acceso en un sistema Linux



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

El usuario **root** en un sistema GNU/Linux es el usuario que tiene acceso administrativo a todo el sistema y puede realizar cualquier tipo de modificación. Este usuario tiene poder y control absoluto sobre el equipo y puede hacer y deshacer a su antojo sin ningún tipo de limitación. Es similar al usuario administrador en Windows, aunque en Linux se tiene mucho más poder de decisión.

En términos de si puede existir más de un usuario con este perfil, técnicamente, solo hay un usuario root en un sistema Linux, identificado por el **UID 0**.

Existe el comando ``sudo`` en Linux, que viene del inglés "super user do", que permite que un administrador del sistema otorgue temporalmente a un usuario estándar o grupos de usuarios la posibilidad de ejecutar algunos o todos los comandos como root, mientras registra todos los comandos y argumentos.

**(d) Agregue un nuevo usuario llamado iso2023 a su instalación de GNU/Linux, especifique que su home sea creada en /home/iso\_2023, 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.**

**PASOS:**

**0-** En primer lugar me di permisos de root a mi usuario principal modificando a mano el archivo `/etc/sudoers` con VI

**1-** Crear el grupo con el comando:

`sudo groupadd catedra`

**2-** Crear el nuevo usuario con el comando

`sudo useradd iso2023`

**3-** Cree un archivo llamado `'ejercicio3.txt'` con el comando:

`sudo touch /home/iso2023/ejercicio3.txt`

**4-** Eliminé el usuar con el comando

`sudo userdel -r iso2023`

Parámetro `-r`:

Files in the user's home directory will be removed along with the home directory itself and the user's mail spool. Files located in other file systems will have to be searched for and deleted manually.

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

**1. useradd / adduser:** Este comando se utiliza para crear un nuevo usuario en el sistema Linux. Los parámetros más importantes incluyen `-d` para especificar el directorio de home del usuario, `-s` para definir la shell de inicio del usuario y `-G` para agregar el usuario a un grupo existente.

**2. usermod:** Este comando se utiliza para modificar la información de inicio de sesión de un usuario existente. Algunas de las opciones más utilizadas incluyen `-a` `-G` para agregar un usuario a un grupo secundario, `-d` para cambiar el directorio home del usuario y `-l` para cambiar el nombre de inicio de sesión del usuario.

**3. userdel:** Este comando se utiliza para eliminar una cuenta de usuario. La opción `-r` se utiliza a menudo con este comando para eliminar también el directorio de inicio del usuario y su correo.

**4. su:** Este comando permite cambiar la identidad del usuario actual a otro usuario. Si se utiliza sin especificar un nombre de usuario, `su` cambia al usuario root.

**5. groupadd:** Este comando se utiliza para crear un nuevo grupo. La opción `-g` permite especificar un GID (ID de grupo) específico al crear un nuevo grupo.

**6. who:** Este comando muestra información sobre los usuarios que están actualmente conectados al sistema. Algunas de las opciones más utilizadas incluyen `-m` para imprimir información solo sobre el terminal asociado con el usuario actual, `-b` para imprimir la hora del último arranque del sistema, `-r` para mostrar el nivel de ejecución actual, `-q` para obtener solo los nombres de usuario y el número de usuarios actualmente conectados

**7. groupdel:** Este comando se utiliza para eliminar un grupo. Solo el root o un usuario con privilegios sudo puede eliminar grupos. Una de las opciones más importantes es `-f` para forzar la eliminación del grupo, incluso si hay algún usuario que tenga el grupo como principa

**8. passwd:** Este comando se utiliza para cambiar la contraseña del usuario. También puede ser utilizado por el root o un usuario con privilegios sudo para cambiar la contraseña de cualquier usuario en el sistema. Algunas de las opciones más utilizadas incluyen `-d` para eliminar la contraseña del usuario y hacer que la cuenta no tenga contraseña, `-e` para hacer que la contraseña de la cuenta expire inmediatamente y obligar al usuario a cambiar la contraseña en su próximo inicio de sesión

## 4. FileSystem:

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

Los permisos sobre archivos en GNU/Linux se definen a través de tres niveles de permisos:

1. **Permisos de propietario:** Es el usuario que creó el archivo o directorio. Cuando creas un archivo, te conviertes en el propietario del mismo.
2. **Permisos de Grupo:** Cada usuario es parte de un grupo(s) determinado(s). Un grupo está formado por varios usuarios y es una forma de gestionar los usuarios en un entorno multiusuario. Contienen múltiples usuarios, lo que hace más fácil el asignar permisos a todos los pertenecientes a dicho grupo.
3. **Otros:** 'Otros' puede considerarse como un supergrupo con todos los usuarios del sistema. Básicamente, cualquier persona con acceso al sistema pertenece a este grupo<sup>1</sup>.

Para cada uno de estos perfiles se define si el usuario puede acceder para leer el archivo, escribir en él o ejecutarlo. Cada archivo y directorio en Linux tiene los siguientes tres permisos para los tres tipos de propietarios, los cuales se basan en una notación octal:

- **Lectura:** Permite a un cierto usuario ver el contenido de un archivo. Si es un directorio, el contenido del mismo.

Valor = R

Octal = 4

- **Escritura:** Permite modificar un archivo. Si es un directorio, el contenido del mismo.

Valor = W

Octal = 2

- **Ejecución:** Si existe algo que ejecutar, se podrá hacer con este permiso.

Valor = X

Octal = 1

Para cambiar los permisos de un archivo, se puede usar el comando ``chmod``. Por ejemplo, si quisieras darle permisos de ejecución al archivo ``hola.txt``, bastaría con ``chmod +x hola.txt``.

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

**chmod:**

Este comando se usa para cambiar los permisos de archivos o directorios en sistemas operativos.

La sintaxis general del comando chmod es:

`chmod opciones permisos "nombre del archivo".`

Los permisos definen los privilegios para el propietario del archivo (el “usuario”), los miembros del grupo que posee el archivo (el “grupo”) y cualquier otra persona (“otros”).

Hay dos formas de representar estos permisos: con símbolos (caracteres alfanuméricos) o con números octales (los dígitos del 0 al 7).

Por ejemplo, se quiere establecer los permisos de la siguiente forma: El usuario puede leer, escribir y ejecutar el archivo. Los miembros del grupo pueden leer y ejecutar el archivo. Otros usuarios solo pueden leer el archivo.

El comando que tendrías que usar sería:

`chmod u=rwx,g=rx,o=r mi-archivo.txt.`

## **chown:**

Este comando se usa para cambiar el propietario y el grupo de archivos, directorios y enlaces.

Con las opciones chown, se puede cambiar la propiedad de los archivos, directorios y enlaces.

Para cambiar el propietario de un archivo, el formato básico del comando es:

```
chown user filename(s).
```

Para cambiar el mismo archivo mi-archivo.txt al propietario galperin y al grupo meli, entonces el comando sería:

```
chown galperin:meli archivo.txt.
```

## **chgrp:**

Este comando nos permite cambiar la propiedad del grupo de un archivo o directorio en el sistema.

Normalmente este tipo de tareas de asignación de permisos se puede realizar con el comando chown pero chgrp maneja una sintaxis más simple para esta tarea.

La sintaxis de uso de chgrp es la siguiente:

```
$ chgrp meli archivo1 [ archivo2 archivo3...]
```

Para cambiar la propiedad del grupo para diversos archivos de forma simultánea debemos hacer uso de comodines, por ejemplo, para cambiar la propiedad de todos los archivos .txt ejecutamos:

```
sudo chgrp meli *.txt.
```

**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?**

La notación octal en el comando chmod se utiliza para establecer los permisos de un archivo o directorio en sistemas operativos Unix y Linux. Cada dígito en la notación octal representa los permisos para el usuario, grupo y otros, **en ese orden**.

Cada dígito es una combinación de los números 4, 2, 1 y 0:

4 significa “leer” (R)

2 significa “escribir” (W)

1 significa “ejecutar” (X)

0 significa “sin permiso”

Por lo tanto, se tiene un número octal como 754 para un archivo:

El primer dígito (7) representa los permisos del propietario.

En este caso, 7 es la suma de 4 (leer), 2 (escribir) y 1 (ejecutar), por lo que el propietario tiene permisos de lectura, escritura y ejecución.

El segundo dígito (5) representa los permisos del grupo. Aquí, 5 es la suma de 4 (leer) y 1 (ejecutar), por lo que el grupo tiene permisos de lectura y ejecución.

El tercer dígito (4) representa los permisos para otros usuarios. El número 4 significa que otros usuarios solo tienen permiso de lectura.

Por lo tanto, `chmod 754 mi-archivo.txt` establecería estos permisos para el archivo

**(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.**

Sí, existe un usuario que puede acceder a cualquier archivo sin importar los permisos establecidos, y ese es el usuario root. El usuario root es el superusuario y tiene privilegios totales sobre el sistema, lo que incluye la capacidad de leer, escribir y ejecutar cualquier archivo. De igual manera, un usuario no root pero con permisos 'sudo' sería capaz de acceder y realizar todas las acciones sin los permisos correspondientes.

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

En los sistemas operativos basados en Unix, como GNU/Linux, los nombres de los archivos y directorios se especifican en cadenas de texto llamadas "rutas" o "paths". Hay dos tipos de rutas: absolutas (o "full path name") y relativas (o "relative path name").

**Ruta Absoluta (Full Path Name):**

Es la ruta que especifica la ubicación de un archivo o directorio desde el directorio raíz (/). En otras palabras, una ruta absoluta es una ruta completa desde el inicio del sistema de archivos desde el directorio '/'. Por ejemplo, si se tiene un archivo llamado 'documento.txt' en el directorio 'Documentos' de tu directorio personal, la ruta absoluta sería '/home/tu\_usuario/Documentos/documento.txt'.

**Ruta Relativa (Relative Path Name):**

Es una ruta que se define en relación con el directorio de trabajo actual. Por ejemplo, si estás actualmente en tu directorio personal ('/home/tu\_usuario') y quieres acceder al mismo archivo 'documento.txt' en el directorio 'Documentos', puedes usar la ruta relativa 'Documentos/documento.txt'.

Las rutas relativas pueden cambiar dependiendo del directorio de trabajo actual, mientras que las rutas absolutas siempre se refieren a la misma ubicación en el sistema de archivos partiendo desde la raíz '/'.



**(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.**

Para determinar en qué directorio te encuentras actualmente en un sistema GNU/Linux, puedes usar el comando ``pwd``, que significa "print working directory" (imprimir directorio de trabajo).

Sí, existe alguna forma de ingresar al directorio personal sin necesidad de escribir todo el path completo. Se puede usar el comando `cd` sin ningún argumento para volver al directorio personal. También se puede usar el comando `cd ~` para el mismo propósito. Ambos comandos llevarán al directorio personal sin necesidad de escribir todo el path completo.

Sí, se puede utilizar la misma idea para acceder a otros directorios. En lugar de escribir la ruta completa, se pueden usar rutas relativas basadas en la ubicación actual.

Por ejemplo, supongamos que estás en tu directorio personal (`/home/tu_usuario``) y tenés un directorio llamado ``Documentos`` dentro de él. Si querés cambiar al directorio ``Documentos``, podés simplemente escribir ``cd Documentos`` en lugar de la ruta completa ``cd /home/tu_usuario/Documentos``.

Además, hay algunos atajos útiles que puedes usar con el comando ``cd``:

- ``cd ..`` te moverá al directorio padre del directorio actual.
- ``cd -`` te llevará al directorio en el que estabas antes.
- ``cd ~`` te llevará a tu directorio personal, sin importar en qué directorio te encuentres actualmente.

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

**cd:** Se utiliza para cambiar el directorio actual en el que te encuentras. Puedes usar `cd ..` para moverte un nivel hacia arriba en la jerarquía de directorios, o `cd [nombre del directorio]` para moverte a un directorio específico.

**umount:** Se utiliza para desmontar sistemas de archivos. Normalmente se utiliza en conjunción con dispositivos de almacenamiento como discos duros.

**mkdir:** Se utiliza para crear un nuevo directorio. Puedes usar `mkdir [nombre del directorio]` para crear un nuevo directorio con el nombre especificado.

**du:** Se utiliza para estimar y mostrar el uso del espacio en disco de archivos y directorios.

**rmdir:** Se utiliza para eliminar (borrar) directorios vacíos.

**df:** Se utiliza para mostrar la cantidad de espacio utilizado y disponible en los sistemas de archivos.

**mount:** Se utiliza para montar sistemas de archivos. Normalmente se utiliza en conjunción con dispositivos de almacenamiento como discos duros, USBs, etc.

**ln:** Se utiliza para crear enlaces a un archivo o directorio existente.

**ls:** Se utiliza para listar los archivos y directorios en el directorio actual.

**pwd:** Se utiliza para imprimir el nombre del directorio de trabajo actual.

**cp:** Se utiliza para copiar archivos y directorios.

**mv:** Se utiliza para mover o renombrar 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 una instancia de un programa en ejecución. Es una unidad de actividad que se caracteriza por la ejecución de una secuencia de instrucciones, un estado actual, y un conjunto de recursos del sistema asociados. Es dinámico.

Las siglas PID y PPID hacen referencia a los identificadores de procesos en un sistema operativo. **PID** es una abreviatura de "process ID", o sea, ID del proceso o bien identificador de procesos. Es un número entero usado por el kernel de algunos sistemas operativos (como el de Unix o el de Windows NT) para identificar un proceso de forma unívoca. Por otro lado, **PPID** son las siglas de "Parent Process ID", lo que significa que Parent Process es el responsable de crear el proceso actual (Child Process).

En GNU/Linux, todos los procesos tienen estos atributos. Cada proceso que se inicia es referenciado con un número de identificación único conocido como Process ID PID, que es siempre un entero positivo. Además de una ID de proceso única, a cada proceso se le asigna una ID de proceso principal (PPID) que indica qué proceso lo inició.

Además del PID y PPID, un proceso en GNU/Linux tiene otros atributos como:

- Nice number: prioridad asignada al ejecutarlo.
- tty: terminal en el que se está ejecutando.
- RUID: identificador del usuario real, el que lo ejecutó.
- EUID: identificador del usuario efectivo.
- RGID: identificador del grupo real.
- EGID: identificador del grupo efectivo.

Estos atributos permiten al sistema operativo gestionar los procesos y asignar los recursos del sistema de manera eficiente.

En la materia, usaremos 'proceso', 'tarea' y 'job' como sinónimos.

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

En un sistema GNU/Linux, hay varios comandos que puedes utilizar para ver los procesos que están en ejecución:

**1. ps:** Este comando puede listar todos los procesos que se están ejecutando en un sistema Linux..

Algunos parámetros son:

- **`ps -e`** para ver todos los procesos actuales.
- **`ps -aux`**, que también muestra el uso actual de la CPU y la RAM de cada proceso, así como el comando que lo ha generado.
- **`ps -u [nombre de usuario]`**: lista todos los procesos en ejecución de un determinado usuario.
- **`ps -T`**: imprime los procesos activos que se ejecutan desde el terminal.
- **`Ps -C nombre\_proceso`**: filtra la lista por el nombre del proceso. Además, este comando también muestra todos los procesos hijos del proceso especificado.

**2. pgrep:** Este comando es una especie de combinación de ps y grep.

Se puede especificar el nombre -o parte del nombre- de un proceso que estés buscando, y pgrep devolverá los respectivos ID de los procesos. Por ejemplo, para buscar cualquier proceso relacionado con SSH en tu sistema, escribirías **`pgrep ssh`**.

**3. top:** El comando top proporciona una vista dinámica en tiempo real del sistema en ejecución, mostrando la información como esperarías verla en una interfaz gráfica.

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

En GNU/Linux, un proceso se dice que se está ejecutando en **Background** (segundo plano) cuando se está ejecutando sin monopolizar la terminal. Esto significa que se puede seguir usando la terminal para otros comandos o tareas mientras el proceso en background continúa su ejecución.

Por otro lado, un proceso se dice que se está ejecutando en **Foreground** (primer plano) cuando se está ejecutando y monopoliza la terminal. Esto significa que no se puede usar la terminal para otros comandos o tareas hasta que el proceso en foreground finalice su ejecución.

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

Si se quiere ejecutar un programa desde la terminal sin que esta quede ocupada por el programa, se puede añadir un `&` detrás del comando. Esto enviará el proceso a segundo plano directamente.

Se puede mover procesos entre el background y el foreground usando los comandos **`bg`** y **`fg`**.

Por ejemplo, se tiene un proceso en segundo plano y se quiere traerlo al primer plano (ocupando la terminal), se puede usar el comando **`fg`** seguido del número de trabajo del proceso. Y si se quiere continuar ejecutando un proceso en segundo plano para que no ocupe la terminal, se ingresaría **`bg`** seguido del número de trabajo del proceso.

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

El Pipe, representado por el símbolo ``|``, es una herramienta poderosa en GNU/Linux. Su finalidad es permitir que los datos fluyan entre dos procesos. El pipe conecta stdout (salida estándar) del primer comando con la stdin (entrada estándar) del segundo. Esto significa que el resultado producido por un programa puede servir como entrada para otro programa. De esta manera, se pueden dividir grandes problemas en problemas más chicos y obtener una mejor visión general.

Ejemplos:

**1. Filtrar archivos en un directorio:** Listar el contenido de un directorio, pero solo te interesa ver los nombres que coinciden con la palabra "doc". Entonces podrías usar un Pipe para canalizar la salida del comando ``ls`` y llevarla a la entrada del filtro ``grep`` para decirle que solo muestre los que coincidan con ese patrón:

```
ls -l | grep doc
```

**2. Filtrar procesos:** Ver la información de los procesos con nombre "firefox" y no todos:

```
ps aux | grep firefox
```

En este caso, en lugar de mostrar toda la salida del programa ``ps`` en la pantalla, lo que hace es canalizarla hacia la entrada del filtro ``grep`` y solo muestra en la salida lo que se corresponde con el patrón "firefox" en este caso.

Se pueden usar varias Pipes para llevar la salida de un comando a la entrada de un segundo comando, y la salida de ese segundo hacia la entrada de un tercero y así sucesivamente. Es decir, se pueden anidar tantos pipes como se deseen.

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

La redirección es una de las características más importantes proporcionadas por la shell. La redirección se refiere a la posibilidad de cambiar la visualización de la pantalla hacia un archivo, una impresora o cualquier otro periférico, los mensajes de errores hacia otro archivo, de sustituir la introducción vía teclado por el contenido de un archivo.

Por defecto, el canal de entrada será siempre el teclado y el canal de salida será siempre la pantalla. Sin embargo, es posible redireccionar estos dos canales hacia un archivo.

**Las redirecciones más comunes son:**

**Redirección de salida destructiva:** Permite enviar el resultado de un comando hacia un archivo. Se utiliza el carácter `>` para redireccionar la salida estándar (la que se muestra por pantalla). Por ejemplo:

```
ls -l > salida
cat salida
```

Si no existe el archivo `salida`, lo creará antes de ejecutar el comando. Si ya existe el archivo, se sobrescribirá todo su contenido.

Para añadir datos a continuación del archivo, se utiliza la

**Redirección de salida no destructiva** `>>`

```
echo "texto al final" >> salida
cat salida
```

**Redirección de entrada:** Los comandos que esperan datos o parámetros desde el teclado también pueden recibirlos con una redirección de entrada utilizando el carácter `<` . Por ejemplo:

```
wc < salida
```

**Redirección de error:** Se puede redireccionar el canal de error a un archivo utilizando `2>` . Por ejemplo:

```
rm noexiste.txt 2> error.log
cat error.log
```

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

El comando `kill` es una herramienta que permite enviar una señal a un proceso. Su principal función es terminar procesos. Cada proceso en un sistema Linux tiene una ID de proceso única, también conocida como PID. El comando `kill` te permite terminar un proceso utilizando este PID.

Por ejemplo, para **terminar un proceso específico**: Para finalizar un proceso específico con un PID, se puede usar el comando:

```
kill 63772
```



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

**ps:**

Este comando proporciona información sobre los procesos que se están ejecutando en el sistema. Puedes usar diferentes opciones para obtener un listado más detallado y específico. Por ejemplo, ``ps -ux`` muestra todos los procesos del usuario actual.

**kill:**

El comando ``kill`` permite enviar una señal a un proceso. Su principal función es terminar procesos. Cada proceso en un sistema Linux tiene una ID de proceso única, también conocida como PID. El comando ``kill`` te permite terminar un proceso utilizando este PID.

**ps tree:**

El comando ``ps tree`` muestra los procesos en ejecución en una estructura de árbol. Esto es útil para visualizar las relaciones entre los procesos.

**killall:**

El comando ``killall`` permite terminar todos los procesos que se están ejecutando con un nombre de comando específico. Si más de un proceso está ejecutando el comando especificado, ``killall`` terminará todos ellos.

**top:**

El comando ``top`` proporciona una visión en tiempo real de los procesos que se están ejecutando en el sistema. Muestra información como el uso de la CPU, la memoria, el tiempo de ejecución y más para cada proceso.

**nice:**

El comando ``nice`` en GNU/Linux permite cambiar la prioridad de un proceso al momento de su creación. Los procesos con mayor prioridad obtienen más tiempo de CPU que los procesos con menor prioridad.

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

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

En el contexto de GNU/Linux, empaquetar se refiere a agrupar varios archivos y/o directorios en un solo archivo.

En Linux, el comando `tar` es comúnmente utilizado para realizar el proceso de empaquetación. Esto se hace frecuentemente cuando se necesita distribuir varios archivos de manera más sencilla, o cuando se quiere comprimir.

Comprimir en el contexto de GNU/Linux se refiere a reducir el tamaño de uno o varios archivos y/o directorios, generalmente con el objetivo de ahorrar espacio en disco o facilitar su transferencia.

### **(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?**

Luego de comprimir los 4 archivos en un `.tar` con el comando `'tar -cf archivo1.xls archivo2.txt [...]'`, noté que el `tar` ocupa un poco más de espacio que los 4 archivos separados.

Esto se debe a que el `.tar` contiene información adicional como los permisos de archivos y las fechas de modificación.

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

Luego de empaquetar los archivos con el comando previamente mencionado en el punto (b), se pueden comprimir los archivos usando el comando `'gzip paquete.tar'`.

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

Sí, es posible comprimir un conjunto de archivos en un solo paso utilizando el comando tar con la opción -z para gzip o -j para bzip2:

```
tar -czf paquete.tar.gz archivo1 archivo2 archivo3 archivo4
```

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

**tar:**

Se utiliza para agrupar varios archivos y/o directorios en un solo archivo.

**grep:**

Se utiliza para buscar un patrón específico en un archivo o grupo de archivos. Este comando es muy versátil y útil, ya que permite buscar una palabra o patrón y se imprimirá la línea o líneas que la contengan.

**gzip:**

Se utiliza para comprimir archivos, reduciendo su tamaño sin perder la calidad ni sus propiedades. Por defecto, gzip cambia el archivo original por uno que posee el mismo nombre pero con la extensión .gz7.

**zgrep:**

Se utiliza para buscar el contenido de un archivo comprimido sin descomprimirlo.

**wc:**

Se utiliza para contar el número de caracteres, palabras y líneas de un archivo. Este comando puede ser utilizado con varios documentos al mismo tiempo, en cuyo caso suma los valores de cada uno de ellos.

**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:**

Lista el contenido del directorio actual y lo guarda en un archivo llamado 'prueba'. Si no existe, lo crea.

**ps > PRUEBA:**

Lista los procesos en ejecución, y lo guarda en un archivo llamado 'PRUEBA.' Si no existe, lo crea.

**chmod 710 prueba:**

Cambia los permisos del archivo prueba. Asigna permisos de lectura, escritura y ejecución al 'propietario' del archivo, permisos solo de ejecución a los miembros del 'grupo', y ningún permiso a 'otros'.

**chown root:root PRUEBA**

Cambia el propietario y el grupo del archivo PRUEBA a root (tanto el propietario como el grupo con acceso).

**chmod 777 PRUEBA**

Cambia los permisos del archivo PRUEBA. Asigna permisos de lectura, escritura y ejecución tanto al propietario del archivo, como a los miembros del 'grupo' y a 'otros'.

**chmod 700 /etc/passwd:**

Le da permisos de lectura, escritura y ejecución al propietario del archivo '/etc/passwd'. Tanto el 'grupo' como 'otros' no tienen ningún permiso.

**passwd root**

Intenta cambiar la contraseña del usuario 'root'. Solo el root puede hacer esta acción.

## **rm PRUEBA**

Borra el archivo PRUEBA. Lo podrá hacer solo si se tiene permisos de escritura sobre el mismo. Como anteriormente este mismo usuario es quien lo creó, no hay problema y lo borra efectivamente.

## **man /etc/shadow**

Intenta acceder al manual de '/etc/shadow'. Solo puede ingresar el usuario 'root'.

## **find / -name \*.conf**

Busca a partir del directorio indicado (en este caso '/', por lo que buscará en todo el filesystem) a los **todos** archivos que en su nombre (-name) contengan **\*.conf**

el \* que es un comodín, es decir en lugar del \* puede ir cualquier cadena de caracteres

## **usermod root -d /home/newroot -L**

Modifica la información del usuario root. En este caso indicado por el parámetro '-d' indica que '/home/newroot' será el nuevo home. Y con -L bloquea la cuenta, no se podría acceder a la cuenta. Como es el root, en realidad si puede.

## **cd /root**

Quiere cambiar el directorio de trabajo actual al directorio **"/root"**. Si el usuario actual no tiene permisos para acceder a este directorio, se generará un error.

## **rm \*:**

Eliminaría todos los archivos del directorio de trabajo actual. Lo más probable es que en el paso anterior haya sido denegado el acceso. En caso de que el usuario que está ejecutando todo esto sea el root, podría efectivamente hacerlo.

## **cd /etc**

Quiere cambiar el directorio de trabajo actual al directorio **"/etc"**.

### **cp \* /home -R**

Copia todo lo del directorio actual ('/etc') a /home de manera recursiva (-R)

### **shutdown**

Apagará o reiniciará el sistema, dependiendo de cómo esté configurado. Este comando generalmente requiere privilegios de root para ejecutarse.

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

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

*kill 23*

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

*sudo kill init*

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

*find /home -name \*.conf*

**(d) Guardar una lista de procesos en ejecución el archivo /home//procesos**

*ps > /home//procesos*

**(e) Cambiar los permisos del archivo /home//xxxx a:  
Usuario: Lectura, escritura, ejecución Grupo: Lectura,  
ejecución Otros: ejecución**

```
chmod 751 /home//xxxx
```

**(f) Cambiar los permisos del archivo /home//yyyy a: Usuario:  
Lectura, escritura. Grupo: Lectura, ejecución Otros: Ninguno**

```
chmod 650 /home//yyyy
```

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

```
rm -f /tmp/*
```

**(h) Cambiar el propietario del archivo /opt/isodata al usuario  
iso2023**

```
chown iso2023 /opt/isodata
```

**(i) Guardar en el archivo /home//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//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 NCaporal  
passwd NCaporal*

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

*Se creo /home/NCaporal  
Se modificaron /etc/passwd y /etc/shadow*

**(d) Crear un directorio en /tmp llamado cursada2021**

*mkdir /tmp/cursada2021*

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

*cp -r /var/log/\* /tmp/cursada2021/*



**(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 -R NCaporal:users /tmp/cursada2021
```

**(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 -R /tmp/cursada2021
```

**(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.**

```
tty
```

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

```
ps | wc -l
```

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

```
who | wc -l
```

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

*wall "Guarde su progreso, el sistema será apagado en 5mins"*

**(m) Apague el sistema**

*shutdown -h*

**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 21322*  
*cd 21322*

**(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".**

*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?.**

```
cd /etc
```

```
ls
```

```
ls /etc > ~/leame
```

~ es igual al directorio personal, es decir, es equivalente a escribir "/home/nicolascaporal/"

Esto se puede realizar sin problemas porque Linux es sensible a mayúsculas y minúsculas, por lo tanto leame y LEAME se consideran archivos diferentes.

**(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.**

Para localizar un archivo dentro del filesystem, se puede usar el comando *find*. Por ejemplo, para buscar un archivo llamado "miarchivo.txt" en todo el sistema, se puede ejecutar:

```
find / -name miarchivo.txt
```

Para buscar varios archivos con características similares, se pueden usar comodines (\*). Por ejemplo, para encontrar todos los archivos que tienen la extensión ".txt", se hace lo siguiente:

```
find / -name "*.txt"
```

Este comando buscará todos los archivos con extensión .txt en todo el sistema

**(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 *.so > ~/21322/ejercicio_f
```

**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 el directorio 'iso' en el directorio de trabajo actual.

**cd . /iso ; ps > f0:**

**Se cambia al directorio 'iso'.**

**Se guarda la lista de procesos actuales en un archivo 'f0'**

El ';' es un separador, para poner 2 comandos en una misma línea, se ejecutan uno después del otro, secuencialmente.

**ls > f1**

Se guarda la lista de archivos del directorio actual en un archivo 'f1'

**cd /**

Se cambia al directorio raíz '/'

**echo \$HOME**

Muestra por consola lo que hay en la variable de entorno '\$HOME'

**ls -l &> \$HOME/iso/ls**

Guarda la lista de archivos del directorio actual en 'ls'.

Con el operador '&>' indica que se redirige tanto la salida estándar como el error estándar. En caso de error con el ls, en vez de informar en consola, lo guarda en el archivo

**cd \$HOME; mkdir f2**

Se cambia al directorio de \$HOME

Y se crea un directorio f2

## **ls -ld f2**

Se lista el directorio f2

(el directorio y no su contenido, eso indica -d)

## **chmod 341 f2**

Se cambiaron los permisos del directorio f2.

Escritura y ejecución para el propietario.

Lectura para el grupo.

Ejecución para otros.

## **touch dir**

Crea un archivo llamado 'dir' en el directorio actual (\$HOME)

## **cd f2**

Cambia al directorio 'f2'

## **cd ~/iso**

Cambia al directorio 'iso' que está en el directorio personal del usuario (~)

## **pwd > f3**

Guarda el directorio de trabajo actual en f3

## **ps | grep 'ps' | wc -l >> ../f2/f3**

De los procesos activos (ps), filtra los que contengan 'ps' (grep), cuenta la cantidad (wc -l) y lo agrega al final (>>) del archivo f3 (que está en el directorio f2, un nivel más arriba en la jerarquía de directorios(../))

## **chmod 700 ../f2 ; cd ..**

Cambia los permisos del directorio f2. Le da todos los permisos al propietario, y ningún permiso para el grupo y otros.

Luego, cambia de directorio, va al padre.

**find . -name etc/passwd**

Busca en el directorio actual (indicado con el '.') y sus subdirectorios archivos que contengan 'etc/passwd'

**find / -name etc/passwd**

Busca en todo el sistema de archivos (indicado con el '/') y sus subdirectorios archivos que contengan 'etc/passwd'

**mkdir ejercicio11**

Crea el directorio 'ejercicio11'

.....

.....

**(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 "ejercicio11\_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 -r ./iso ~/ejercicio11
```

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

```
cp -nr * ~/ejercicio11
```

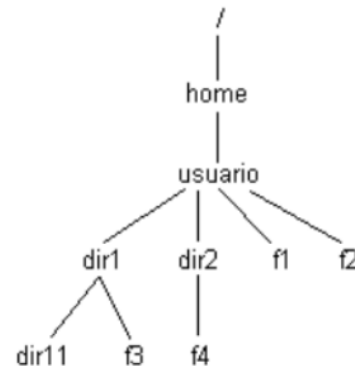
Con \* copio todos los archivos.

Usando '-n' evito que sobrescriba los que ya estaban.

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

**di12. 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:**

\*Los comandos en todos los casos están lanzados desde /usuario.  
Entre paso y paso hay un “cd ~”

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

```
mv dir1/f3 ~
```

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

```
cp dir2/f4 dir1/dir11
```

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

```
cp dir2/f4 dir1/f7
```

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

```
mkdir ~/copia ; cp -r ~/dir1 ~/copia
```

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

```
mv f1 archivo ; ls -l 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" "f4" de manera que se llamen "f3.exe" y "f4.exe" respectivamente.**

```
mv f3 f3.exe ; cd dir1/dir11 ; mv 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 ~/dir1/dir11/f4.exe ~/f3.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.**

```
mkdir /tmp/logs
```

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

```
sudo cp -r /var/log/* /tmp/logs
```

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

```
sudo tar -cf misLogs.tar /tmp/logs
```

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

```
sudo tar -czf misLogs.tar.gz /tmp/logs
```

**(e) Copie los archivos creados en (c) y (d) al directorio de trabajo de su usuario.**

```
cp misLogs.tar misLogs.tar.gz ~
```

ya estaban en ~

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

```
rm -r /tmp/logs
```

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

```
tar -xf ~/misLogs.tar -C ~/logs1dir
```

```
tar -xzf ~/misLogs.tar.gz -C ~/logs2dir
```