Practica 2

1. Editor de textos:
   (a) Nombre al menos 3 editores de texto que puede utilizar desde la línea de comandos.
   (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.
   (c) Nombre los comandos más comunes que se le pueden enviar al editor de textos vi.

a. vim, emacs, joe

b. Diferencias entre un editor de texto y los comandos **cat**, **more** o **less**:

   Editor de texto:

   - Un editor de texto, como **vi**, permite **crear, editar y guardar archivos de texto**.
   - Ofrece **herramientas interactivas** para manipular el contenido del archivo (cortar, copiar, pegar, buscar, reemplazar).
   - Tiene una **interfaz interactiva** donde puedes moverte dentro del archivo y hacer cambios directamente.
   - Los cambios se **pueden guardar o descartar**.

   Comandos cat, more, y less:

   - **cat**: Solo muestra el contenido de un archivo en la terminal. No permite edición.
   - **more**: Permite ver un archivo **página por página** en la terminal, pero no permite editar.
   - **less**: Similar a **more**, pero más avanzado, permitiendo la **navegación bidireccional** (hacia adelante y atrás). Tampoco permite editar.
   - **Estos comandos no permiten modificar el archivo**; solo están diseñados para visualizarlo.

Modos de operacion del editor vi

1. **Modo Normal**: Es el modo por defecto cuando se abre **vi**. Aquí se puede **navegar** por el texto, eliminar líneas o palabras, copiar y pegar texto. No se pueden hacer ediciones directas de texto.
2. **Modo de Inserción**: En este modo puedes **insertar y editar texto**. Se activa con comandos como `i` (insertar) o `a` (añadir texto).
3. **Modo de Línea de Comandos**: Permite ejecutar **comandos especiales** como guardar (`:w`), salir (`:q`), buscar (`:/`) y reemplazar. Se accede usando los dos puntos `:`.
4. **Modo Visual**: Permite **seleccionar bloques de texto** para copiarlos, eliminarlos o aplicar otras acciones. Se activa con `v`, `V` o `Ctrl+v` (visual block).

c. Comandos mas comunes:

- w: escribir cambios
- q o q!: salir del editor
- dd: cortar
- y: copiar al portapapeles
- p: pegar desde el portapapeles
- u: deshacer
- /frase: busca “frase” dentro del archivo

2. Proceso de Arranque SystemV (https://github.com/systeminit/si):
(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.
(b) Proceso INIT. ¿Quién lo ejecuta? ¿Cuál es su objetivo?
(c) RunLevels. ¿Qué son? ¿Cuál es su objetivo?
(d) ¿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?
(e) Archivo /etc/inittab. ¿Cuál es su finalidad? ¿Qué tipo de información se

almacena en
el? ¿Cuál es la estructura de la información que en él se almacena?
(f) Suponga que se encuentra en el runlevel <X>. Indique qué comando(s) ejecutaría para
cambiar al runlevel <Y>. ¿Este cambio es permanente? ¿Por qué?
(g) 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.

a. Pasos del proceso de inicio de un sistema GNU/Linux:
   - Se empieza a ejecutar el codigo del BIOS
   - El BIOS ejecuta el POST
   - El BIOS lee el sector de arranque (MBR)
   - Se carga el gestor de arranque (MBC)
   - El bootloader carga el kernel y el initrd
   - Se monta el initrd como sistema de archivos raiz y se inicializan componentes esenciales (ej.: scheduler)
   - El Kernel ejecuta el proceso init y se desmonta el initrd
   - Se lee el /etc/inittab
   - Se ejecutan los scripts apuntados por el runlevel 1
   - El final del runlevel 1 le indica que vaya al runlevel por defecto
   - Se ejecutan los scripts apuntados por el runlevel por defecto
   - El sistema esta listo para usarse

b. El proceso **INIT** es ejecutado por el **kernel** de Linux una vez que ha terminado de cargarse y ha montado el sistema de archivos raíz. Es el primer proceso que se inicia en el sistema operativo y siempre tiene el **PID 1** (Proceso ID 1). En los sistemas modernos, el rol del proceso INIT generalmente lo desempeña **systemd** (aunque en versiones más antiguas o en distribuciones específicas aún se utiliza **SysVinit** o alternativas como **Upstart** o **OpenRC**).

¿Cuál es su objetivo?

El objetivo principal del proceso **INIT** es inicializar el sistema después de que el kernel ha terminado su tarea. Esto incluye:

- **Lanzar y gestionar servicios y daemons** esenciales para que el sistema funcione correctamente, como el servidor de red, la administración de discos, la seguridad, etc.
- **Establecer los niveles de ejecución** o, en el caso de systemd, los **targets** que determinan qué servicios y procesos deben ejecutarse en cada modo de operación (por ejemplo, solo consola, modo gráfico, mantenimiento).
- **Mantener el ciclo de vida de los procesos hijos**, asegurándose de que los procesos que maneja funcionen de forma continua o sean reiniciados en caso de fallo.
- **Gestionar el apagado y reinicio del sistema**, ya que INIT es el proceso encargado de realizar estos procedimientos de manera ordenada y segura.

c. son diferentes estados predefinidos del sistema, cada uno configurado para iniciar un conjunto específico de servicios y procesos. Su objetivo es proporcionar diferentes modos de operación del sistema, desde un modo básico de mantenimiento (single-user mode) hasta un entorno multiusuario con soporte gráfico completo.

d. **RunLevels**

- `0`: Apagar el sistema.
- `1`: Modo de usuario único (rescate).
- `2`: Modo multiusuario sin red (en algunas distribuciones).
- `3`: Modo multiusuario con red, sin entorno gráfico.
- `4`: No utilizado en muchas distribuciones (personalizable).
- `5`: Modo multiusuario con entorno gráfico.
- `6`: Reiniciar el sistema.

El runlevel predeterminado que se utiliza al arrancar el sistema está definido en el archivo de configuración de **SysVinit**, generalmente ubicado en `/etc/inittab`.

En este archivo, se encuentra una línea que indica cuál es el runlevel por defecto. No todas las distribuciones respetan de forma estricta estos estándares, aunque la mayoría sigue la convención general de los runlevels tradicionales.

e. **Finalidad**

El archivo `/etc/inittab` es una parte fundamental de los sistemas que utilizan el sistema de inicialización **SysVinit** (que ha sido reemplazado en muchos casos por **systemd**). Su principal finalidad es definir el comportamiento del proceso de inicialización del sistema, gestionando aspectos como:

- El **runlevel** predeterminado en el que el sistema debe arrancar.
- Los procesos y scripts que se deben ejecutar al cambiar de runlevel.
- Los procesos críticos del sistema, como la gestión de la consola virtual y los terminales.

Tipo de información almacenada

El archivo `/etc/inittab` contiene información sobre:

1. **Runlevel predeterminado**: El runlevel en el que el sistema debe arrancar.
2. **Terminales**: Configuración de terminales y consolas virtuales.
3. **Procesos**: Procesos o servicios que deben iniciarse al entrar en un runlevel específico.
4. **Acciones especiales**: Definiciones de procesos que se deben ejecutar en eventos específicos (como reinicios o apagados).

Estructura de la información

Cada línea en el archivo sigue una estructura de cuatro campos separados por dos puntos (`:`). La estructura general es:

```
id:runlevels:action:process
```

Donde:

- `id`: Es un identificador único para la entrada, generalmente un nombre corto que describe la tarea o terminal asociada (por ejemplo, `1`, `2`, `tty1`, etc.).

- `runlevels`: Define los runlevels en los que se debe ejecutar el comando o proceso. Puede ser un solo número (como `3` o `5`) o una lista de niveles (por ejemplo, `12345`).
- `action`: Especifica la acción que el sistema debe tomar en ese runlevel. Algunas de las acciones más comunes son:
  - `initdefault`: Define el runlevel por defecto.
  - `sysinit`: Se ejecuta una sola vez al inicio del sistema.
  - `wait`: Ejecuta el proceso una vez cuando se entra en el runlevel y espera a que termine.
  - `respawn`: Si el proceso termina, se reinicia automáticamente.
  - `ctrlaltdel`: Define el comportamiento cuando se pulsa la combinación de teclas `Ctrl+Alt+Del`.
- `process`: El comando o proceso que debe ejecutarse. Puede ser un demonio, script, o cualquier otro comando del sistema.

f. Para cambiar entre runLevels podemos utilizar los comandos init o telinit. Estos cambios se hacen de manera temporal hasta el proximo reinicio de la maquina. Para realizar un
cambio permanente, deberiamos modificar el archivo /etc/inittab.

g. Los scrpts RC tienen un papel crucial en la administracion de servicios y procesos durante el arranque y la deteccion del sistema. En el sistem V se almacenan generalmente en los directorios /etc/init.d/ y en los subdirectorios especificos para cada runLevel, ubicados en /etc/rc.d/.
Cuando el sistema arranca o se apaga, init se encarga de gestionar los runLevels consultando el directorio correspondiente para el runLevel actual y ejecuta los scripts segun el orden definido en esos directorios.

3. SystemD(https://github.com/systemd/systemd):
   (a) ¿Qué es sytemd?
   (b) ¿A qué hace referencia el concepto de Unit en SystemD?
   (c) ¿Para que sirve el comando systemctl en SystemD?
   (d) ¿A qué hace referencia el concepto de target en SystemD?

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

a. Es un sistema que centraliza la administracion de demonios (procesos en segundo plano) y
librerias del sistema:
   - Mejora el paralelismo de booteo
   - 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 mas
b. El concepto de Unit en System D hace referencia a las unidades de trabajo
c. El comando `systemctl` es una herramienta fundamental en **systemd**, el sistema de inicialización moderno utilizado en muchas distribuciones GNU/Linux. `systemctl` permite gestionar los **servicios**, **unidades** (units), **targets**, y otros componentes del sistema de manera flexible y eficiente.
d. En **systemd**, un **target** es un mecanismo utilizado para agrupar y organizar unidades (services, sockets, mounts, etc.) que deben ejecutarse o estar activas en un estado específico del sistema. Los **targets** sirven como puntos de sincronización para gestionar los diferentes estados en los que puede estar el sistema, como el inicio, el modo gráfico, el apagado, etc.
e. Una jerarquia de directorios del SystemD

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

   (b) ¿A qué hacen referencia las siglas UID y GID? ¿Pueden coexistir UIDs iguales en un

sistema GNU/Linux? Justifique.
(c) ¿Qué es el usuario root? ¿Puede existir más de un usuario con este perfil en GNU/Linux? ¿Cuál es la UID del root?.
(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.
(e) Investigue la funcionalidad y parámetros de los siguientes comandos:

- useradd ó adduser
- usermod
- userdel
- su
- groupadd
- who
- groupdel
- passwd

a. Archivos utilizados para guardar la informacion

etc/passwd

$ cat /etc/passwd
ndelrio:x:2375:500:Nico del Rio,,,,Usuarios:/
home/admins/ndelrio:/bin/bash

/etc/group

$ cat /etc/group
infraestructura:x:500:

/etc/shadow

```
$ cat /etc/shadow
ndelrio:$1$HamkgCYM$TtgfLJLpIltxutaiqh/u9
/:13273:0:99999:7:::
```

b. **UID (User ID)**: Es un número único asignado a cada usuario en el sistema GNU/Linux. El UID identifica de forma única a cada usuario en el sistema. Los UID están asociados con las cuentas de usuario y se utilizan en lugar del nombre de usuario en las operaciones internas del sistema.

GID (Group ID): Es un número único asignado a cada grupo en el sistema GNU/Linux. El GID identifica de forma única a los grupos de usuarios. Los GID están asociados con los grupos de usuarios para gestionar permisos y accesos compartidos.

Sí, pero no es recomendable. Técnicamente, es posible tener múltiples usuarios con el mismo **UID** en un sistema GNU/Linux, pero puede generar problemas y confusión. Esto sucede porque los permisos y las propiedades de los archivos y procesos se basan en el **UID**, no en el nombre de usuario.

c. **Usuario root** es el usuario con privilegios de superusuario en sistemas GNU/Linux y otros sistemas Unix-like. Este usuario tiene acceso completo y sin restricciones a todos los archivos, comandos y recursos del sistema. El usuario root puede realizar cualquier operación en el sistema, incluidas la administración de usuarios, la configuración del sistema y la instalación de software.

▼ **¿Puede existir más de un usuario con este perfil en GNU/Linux?**

Sí, puede existir más de un usuario con privilegios de superusuario en GNU/Linux, pero esto debe manejarse con cuidado.

El **UID (User ID)** del usuario root es **0**. Este UID especial indica que el usuario tiene privilegios totales en el sistema.

d.

1. Crear el grupo `catedra` (si no existe)

Primero, verifica si el grupo `catedra` ya existe:

```
getent group catedra
```

Si no existe, créalo con el siguiente comando:

```
sudo groupadd catedra
```

2. Agregar el nuevo usuario `iso2017`

Ahora, crea el nuevo usuario `iso2017`, especifica el directorio home como `/home/iso_2017`, y añádelo al grupo `catedra`:

```
sudo useradd -m -d /home/iso_2017 -g catedra iso2017
```

- `m`: Crea el directorio home si no existe.
- `d /home/iso_2017`: Especifica la ubicación del directorio home.
- `g catedra`: Asocia al usuario con el grupo `catedra`.

3. Asignar una contraseña al nuevo usuario

Asigna una contraseña al usuario `iso2017`:

```
sudo passwd iso2017
```

4. Crear un archivo en el home del usuario `iso2017`

Para crear un archivo en el directorio home del usuario sin iniciar sesión como ese usuario, puedes usar el comando `touch` y cambiar la propiedad del archivo:

```
sudo touch /home/iso_2017/archivo_prueba.txt
sudo chown iso2017:catedra /home/iso_2017/archivo_prueba.tx
t
```

- `touch /home/iso_2017/archivo_prueba.txt`: Crea un archivo vacío.
- `chown iso2017:catedra /home/iso_2017/archivo_prueba.txt`: Cambia el propietario y grupo del archivo al usuario `iso2017` y grupo `catedra`.

5. Eliminar el usuario `iso2017`

Para eliminar el usuario `iso2017` y su directorio home:

```
sudo userdel -r iso2017
```

- `r`: Elimina el directorio home del usuario junto con sus archivos.

6. Verificar que no queden registros del usuario

Verifica que no haya registros del usuario en los archivos de información:

- **En** `/etc/passwd`: Este archivo contiene información sobre las cuentas de usuario.

  ```
  grep iso2017 /etc/passwd
  ```

  Si no hay salida, el usuario ha sido eliminado correctamente.

- **En** `/etc/shadow`: Este archivo contiene las contraseñas encriptadas y detalles de las contraseñas.

  ```
  grep iso2017 /etc/shadow
  ```

  Si no hay salida, el usuario ha sido eliminado correctamente.

- **En** `/etc/group`: Este archivo contiene información sobre los grupos.

  ```
  grep iso2017 /etc/group
  ```

  Si no hay salida, el usuario ya no está listado en ningún grupo.

- **En** `/etc/gshadow`: Este archivo contiene información de grupos y contraseñas de grupo.

  ```
  grep iso2017 /etc/gshadow
  ```

  Si no hay salida, el usuario ha sido eliminado de todos los grupos.

e.

- useradd <nombreUsuario>:
  • Agrega el usuario

- Modifica los archivos /etc/passwd
- Alternativa → adduser

- passwd <nombreUsuario>:
  - Asigna o cambia la contraseña del usuario
  - Modifica el archivo /etc/shadow
- usermod <nombreUsuario>:
  - -g: modifica grupo de login (Modifica /etc/passwd)
  - -G: modifica grupos adicionales (Modifica /etc/group)
  - -d: modifica el directorio home (Modifica /etc/passwd)
- userdel <nombreUsuario>: elimina el usuario
- groupdel <nombreGrupo>: elimina el grupo
- su <opciones><usuario>
  - permite cambiar de usuario en una sesión de terminal sin cerrar la sesión actual
- groupadd<opciones><nombreGrupo>
  - se utiliza para crear nuevos grupos en el sistema.
- who<opciones>
  - muestra información sobre los usuarios que están actualmente conectados al sistema.

5. FileSystem:
   (a) ¿Cómo son definidos los permisos sobre archivos en un sistema GNU/Linux?
   (b) Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con
   los permisos en GNU/Linux:
   chmod
   chown
   chgrp
   (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?
   (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.
(e) Explique los conceptos de “full path name” y “relative path name”. De ejemplos claros
de cada uno de ellos.
(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.
(g) Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con
el uso del FileSystem

- cd
- umount
- mkdir
- du
- rmdir
- df
- mount
- ln
- ls
- pwd
- cp
- mv

a. En GNU/Linux, los permisos sobre archivos y directorios se gestionan mediante un sistema que define qué usuarios pueden realizar ciertas acciones (lectura, escritura, y ejecución) sobre ellos. Los permisos están organizados en tres niveles:

a. **Usuario (Owner)**: El propietario del archivo.

b. **Grupo (Group)**: Un grupo de usuarios que pueden tener permisos específicos sobre el archivo.

c. **Otros (Others)**: Cualquier otro usuario que no sea ni el propietario ni parte del grupo.

Cada archivo o directorio tiene tres tipos de permisos:

- **r (read)**: Permiso de lectura (4).
- **w (write)**: Permiso de escritura (2).
- **x (execute)**: Permiso de ejecución (1).

Los permisos se pueden ver y cambiar utilizando el comando `ls -l`, que muestra una cadena de 10 caracteres. Por ejemplo, la salida:

```
-rwxr-xr--
```

se interpreta como:

- El primer carácter (`-`) indica si es un archivo regular o directorio (`d`).
- Los siguientes tres caracteres (`rwx`) son los permisos del **usuario** (lectura, escritura, ejecución).
- Los siguientes tres (`r-x`) son los permisos para el **grupo**.
- Los últimos tres (`r--`) son los permisos para **otros**.

b. chmod → se utiliza para cambiar los permisos de archivos y directorios en GNU/Linux. Los permisos incluyen leer, escribir y ejecutar, y se aplican a tres categorías: usuario (owner), grupo y otros. Sintaxis:

```
chmod [opciones] modo archivo
```

chown → se utiliza para cambiar el propietario y/o grupo de un archivo o directorio. Sintaxis:

```
chown [opciones] propietario[:grupo] archivo
```

chgrp → se utiliza para cambiar el grupo de un archivo o directorio. Sintaxis:

```
chgrp [opciones] grupo archivo
```

c. Existen 3 tipos de permisos y se basan en una notacion octal

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

Se aplican sobre los usuarios:

- Usuario: permisos del dueño → U
- Usuario: permisos del grupo → G
- Usuario: permisos de otros usuario → O

Se utiliza el comando **chmod**:

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

d. Si, es posible:

- **Acceso como superusuario (root)**:
  El superusuario (root) tiene permisos sobre todos los archivos del sistema y puede acceder a cualquier archivo, sin importar las restricciones de permisos.
- **suid o sgid**:
  Si un archivo ejecutable tiene el bit
  suid (set user ID) o **sgid** (set group ID) activado, puede ejecutarse con los permisos del propietario del archivo o del grupo, lo que potencialmente puede dar acceso a recursos a los que el usuario que ejecuta el archivo no tiene permisos directos.

e. Full path name (ruta absoluta):

Es una ruta que comienza desde la raíz del sistema de archivos (el directorio raíz `/`). Siempre especifica la ubicación exacta de un archivo o directorio, sin importar la ubicación actual en la que te encuentres. Ejemplo:

```
/home/usuario/documentos/documento.txt
```

Relative path name (ruta relativa):

Es una ruta que se especifica en relación con el directorio actual en el que te encuentras (conocido como el **directorio de trabajo actual** o **current working directory**). No empieza con `/`, y su interpretación depende de la ubicación desde la cual estás ejecutando comandos. Ejemplo:

```
documentos/documento.txt
```

f. Para determinar en que directorio te encontras actualmente se usa el siguiente comando:

```
pwd
```

Si, es posible, de la siguiente manera:

```
cd ~
```

f. Funcionalidad y parametros de los siguientes comandos:

- cd → Cambia el directorio actual

```
cd home/Desktop/2do_año/ISO
```

- umount → Desmonta un sistema de archivos o dispositivo que fue montado con `mount`.

```
umount [dispositivo | punto de montaje]
```

- mkdir → Crea uno o mas directorios

```
mkdir [opciones] nombre de directorio
```

Parametros:

- -p → Crea directorios anidados (si no existen)

```
mkdir -p /ruta/a/crear/varios/directorios
```

- du → Muestra el espacio de disco utilizado por archivos y directorios

Parametros:

- -h → Muestra el tamaño en un formato legible para humanos (KB, MB, GB).
- -s → Muestra un resumen total en lugar de todos los archivos.
- -a → incluye archivos ademas de los directorios

```
du [opciones] [archivo/directorio]
```

- rmdir → Elimina un directorio vacio

```
rmdir directorio
```

- df → Muestra información sobre el espacio en disco disponible y usado en los sistemas de archivos.

Parametros:

- -h → Muestra el espacio legible en un formato para humanos
- -T → Muestra el tipo de sistemas de archivos

```
df [opciones]
```

- mount → Monta un sistema de archivos en un directorio, permitiendo acceder al contenido del dispositivo.

Parametros:

- -o → Opciones adicionales (como `ro` para montar en modo de solo lectura).

- -t → Tipo de sistema de archivos (por ejemplo, `ext4`, `vfat`).

```
mount [opciones] dispositivo punto_de_montaje
```

- ln → Crea enlaces (hard links o soft links) a archivos o directorios.

Parametros:

- -s → Crea un enlace simbolico (soft link)

```
ln [opciones] archivo enlace
```

- ls → Lista el contenido de un directorio

Parametros:

- -l → Muestra la lista en formato detallado
- -a → Muestra archivos ocultos
- -h → Muestra el tamaño de archivos en formato legible para humanos

```
ls [opciones] [directorio]
```

- pwd → Muestra la ruta absoluta del directorio de trabajo actual

```
pwd
```

- cp → Copia archivos o directorios

Parametros:

- -r → Copia recursivamente (para directorios
- -v → Muestra detalles sobre el proceso de copia
- -i → Pregunta antes de sobreescribir archivos existentes

```
cp [opciones] origen destino
```

- mv → Mueve o renombra archivos y directorios

Parametros:

- -i → Pregunta antes de sobreescribir archivos
- -v → Muestra detalles sobre el proceso de movimiento

```
mv [opciones] origen destino
```

6. 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.
   (b) Indique qué comandos se podrían utilizar para ver qué procesos están en ejecución en
   un sistema GNU/Linux.
   (c) ¿Qué significa que un proceso se está ejecutando en Background? ¿Y en Foreground?
   (d) ¿Cómo puedo hacer para ejecutar un proceso en Background? ¿Como puedo hacer para
   pasar un proceso de background a foreground y viceversa?
   (e) Pipe (|). ¿Cuál es su finalidad? Cite ejemplos de su utilización.
   (f) Redirección. ¿Qué tipo de redirecciones existen? ¿Cuál es su finalidad? Cite ejemplos
   de utilización.
   (g) Comando kill. ¿Cuál es su funcionalidad? Cite ejemplos.
   (h) Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con
   el manejo de procesos en GNU/Linux. Además, compárelos entre ellos:
   ps
   kill
   pstree
   killall
   top
   nice

a. Un **proceso** en un sistema operativo GNU/Linux es una instancia en ejecución de un programa. Un proceso puede ser cualquier tipo de tarea

que el sistema esté ejecutando, ya sea una aplicación interactiva, un comando del sistema, o incluso un script en segundo plano. Cada proceso tiene su propio espacio de direcciones, recursos del sistema (como CPU, memoria) y atributos, lo que permite que el sistema operativo gestione múltiples procesos simultáneamente.

PID:

Es el identificador único que el sistema asigna a cada proceso en ejecución. Es un número entero positivo que permite al sistema operativo y a otros procesos identificar y gestionar el proceso.

PPID:

Es el identificador del proceso padre, es decir, el proceso que creó al proceso actual. Cada proceso en Linux es creado por otro proceso, por lo tanto, el PPID hace referencia al proceso que originó el actual.

▼ ¿Todos los procesos tienen estos atributos en GNU/Linux?

Sí, todos los procesos en GNU/Linux tienen un **PID** y un **PPID**. El PID es fundamental para identificar cada proceso de manera única en el sistema, y el PPID permite organizar la jerarquía de procesos, ya que todos los procesos en Linux (excepto el proceso inicial) son creados por un proceso padre.

b. Comandos que se pueden utilizar para ver los procesos que estan en ejecucion:

- ps
- top
- htop
- pgrep
- pidof
- jobs
- watch
- systemctl
- killall

c. Proceso en Foreground (Primer plano)

Un proceso en **foreground** o primer plano es aquel que se está ejecutando directamente en la terminal o consola que el usuario está utilizando. Mientras un proceso está en primer plano, la terminal no puede aceptar otros comandos hasta que el proceso termine su ejecución o sea enviado al segundo plano (background).

Proceso en Background (segundo plano)

Un proceso en **background** o segundo plano es aquel que se ejecuta "detrás de escena", permitiendo que la terminal acepte otros comandos mientras el proceso sigue ejecutándose en paralelo. Los procesos en segundo plano no requieren interacción directa del usuario ni bloquean la terminal.

d. Para ejecutar un proceso directamente en **background** (segundo plano), simplemente debes añadir el carácter `&` al final del comando que deseas ejecutar. Esto le indica al sistema que inicie el proceso y devuelva inmediatamente el control a la terminal, mientras el proceso sigue ejecutándose en segundo plano. Ejemplo:

```
firefox &
```

Enviar un proceso de Foreground a Background

- **Suspender el proceso**: Presionar `Ctrl + Z`. Esto pausara temporalmente el proceso y lo deja en estado suspendido.
- **Enviar el proceso al background**: Una vez suspendido, usa el comando `bg` para reanudar el proceso, pero en segundo plano:

Traer un proceso de Background a Foreground

Si tienes un proceso ejecutándose en segundo plano y deseas traerlo de vuelta al primer plano, usa el comando `fg`. Si solo tienes un proceso en segundo plano, puedes ejecutarlo así:

```
fg
```

e. El “|” nos permite comunicar dos procesos por medio de unpipe o tuberia desde la shell. El pipe conecta stdout(salida estandar) del primer comando con la stdin (entrada estandar) del segundo. Por ejemplo:

```
$ ls | more
```

> Se ejecuta el comando ls y la salida del mismo es enviada como entrada del comando more

Se pueden anidar tantos pipes como se deseen.

▼ ¿Como hariamos si quisieramos 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. Redireccion

En GNU/Linux y otros sistemas Unix-like, la redirección de entrada y salida es una técnica que permite dirigir el flujo de datos entre los comandos y los archivos. Esta técnica es muy útil para guardar resultados en archivos, leer datos desde archivos, y controlar el flujo de datos en la línea de comandos.

Tipos de redireccion:

- 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 informacion al final

h. Funcionalidad y parametros de los siguientes comandos:

- ps → Muestra informacion sobre los procesos en ejecucion en el sistema

Parametros:

- ps → Muestra los procesos asociados a la sesion actual
- ps aux → Muestra todos los procesos del sistema, con informacion detallada
- ps -ef → Muestra una lista completa de procesos con formato extendido

```
ps aux
```

- kill → Envia señales a procesos, normalmente para terminarlos

Parametros:

- kill PID → Envia la señal SIGTERM (terminacion) al proceso con el PID especificado.
- kill -9 PID → Envia la señal SIGKILL (terminacion forzada) al proceso con el PID especificado.
- kill -s SIGNAL PID → Envia una señal especifica al proceso

```
kill 1234
```

- pstree → Muestra los procesos en ejecucion en una estructura de arbol, mostrando la jerarquia entre procesos padres e hijos

Parametros:

- pstree → Muestra los procesos en un formato de arbol
- pstree -p → Muestra el PID de cada proceso en el arbol
- pstree -u → Muestra el nombre del usuario que ejecuta cada proceso

```
pstree -p
```

- killall → Envia señales a todos los procesos que coinciden con un nombre especifico

Parametros:

- killal nombre_proceso → Envia la señal SIGTERM a todos los procesos con el nombre especificado
- killall -9 nombre_proceso → Envia la señal SIGKILL a todos los procesos con el nombre especificado

```
killall firefox
```

- top → Muestra una lista en tiempo real de los procesos en ejecucion, actualizandose periodicamente

Parametros:

- top → Muestra la lista de procesos en tiempo real
- top -d intervalo → Cambia el intervalo de actualizacion en segundos

```
top
```

- nice → Ejecuta un comando con una prioridad de CPU modificada

Parametros:

- nice -n prioridad comando → Ejecuta el comando con la prioridad especificada. La prioridad puede ser un número entre -20 (máxima prioridad) y 19 (mínima prioridad).
- renice -n prioridad -p PID → Cambia la prioridad de un proceso en ejecución.

```
nice -n 10 command
```

Comparacion entre los comandos

- `ps` **vs** `top`: Ambos muestran información sobre los procesos, pero `ps` proporciona una instantánea estática, mientras que `top` muestra una lista dinámica que se actualiza en tiempo real.
- `ps` **vs** `pstree`: `ps` muestra procesos en un formato de lista, mientras que `pstree` muestra la jerarquía de procesos en forma de árbol.
- `kill` **vs** `killall`: `kill` envía señales a procesos específicos usando su PID, mientras que `killall` envía señales a todos los procesos con un nombre específico.
- `kill` **vs** `nice`: `kill` se utiliza para enviar señales a procesos, normalmente para terminarlos, mientras que `nice` se utiliza para ajustar la prioridad de CPU de un proceso.

- `top` **vs** `pstree`: `top` proporciona una vista en tiempo real de los procesos y su uso de recursos, mientras que `pstree` muestra la jerarquía de los procesos en un formato de árbol.
- `nice` **vs** `top`: `nice` se utiliza para ajustar la prioridad de ejecución de un comando, mientras que `top` monitorea los procesos y su uso de recursos en tiempo real.

7. Otros comandos de Linux (Indique funcionalidad y parámetros):
   (a) ¿A qué hace referencia el concepto de empaquetar archivos en GNU/Linux?
   (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?
   (c) ¿Qué acciones debe llevar a cabo para comprimir 4 archivos en uno solo? Indique la
   secuencia de comandos ejecutados.
   (d) ¿Pueden comprimirse un conjunto de archivos utilizando un único comando?
   (e) Investigue la funcionalidad de los siguientes comandos:
   tar
   grep
   gzip
   zgrep
   wc

a. En GNU/Linux, empaquetar archivos se refiere al proceso de combinar múltiples archivos y/o directorios en un solo archivo, conocido como paquete. Este proceso simplifica la gestión y distribución de archivos.

b y c. **Seleccionar y sumar el tamaño de los archivos**:

Primero, navega al directorio donde están los archivos:

```
cd mi_directorio
```

Luego, usa el comando `du` (disk usage) para ver el tamaño de los archivos. Supongamos que los archivos se llaman `archivo1.txt`, `archivo2.txt`, `archivo3.txt`, y `archivo4.txt`:

```
du -sh archivo1.txt archivo2.txt archivo3.txt archivo4.txt
```

Esto te dará el tamaño de cada archivo en un formato legible por humanos.

Para sumar los tamaños, puedes usar el comando `du` combinado con `awk`:

```
du -b archivo1.txt archivo2.txt archivo3.txt archivo4.txt |
awk '{sum += $1} END {print sum}'
```

Este comando sumará los tamaños en bytes y te dará el total.

- **Crear un archivo empaquetado**:

  Usa el comando `tar` para empaquetar los archivos en un archivo `.tar`. Por ejemplo:

  ```
  tar -cvf archivos.tar archivo1.txt archivo2.txt archivo
  3.txt archivo4.txt
  ```

  Este comando crea un archivo `archivos.tar` que contiene los cuatro archivos seleccionados.

- **Comparar tamaños**:

  Para comparar el tamaño del archivo empaquetado con la suma de los tamaños de los archivos individuales, primero verifica el tamaño del archivo empaquetado:

  ```
  du -sh archivos.tar
  ```

  Luego, compara el tamaño del archivo `.tar` con la suma de los tamaños individuales que calculaste antes.

d. Si se puede. Ejemplo en el inciso c.

e. Funcionalidad de los comandos:

1. **tar**: Es utilizado para empaquetar archivos en un solo archivo (tarball) y, opcionalmente, comprimirlos. Algunos parámetros útiles son:
   - -c: Crear un archivo tar.
   - -v: Mostrar detalles del proceso.
   - -f: Especificar el nombre del archivo tar.
   - -z: Comprimir el archivo tar con **gzip**.
   - -x: Extraer archivos de un archivo tar.
2. **grep**: Busca patrones en el contenido de los archivos. Ejemplo:
   - `grep "palabra" archivo.txt`: Busca la palabra "palabra" en el archivo.
   - Parámetros comunes:
     - -i: Ignora mayúsculas y minúsculas.
     - -r: Busca recursivamente en directorios.
3. **gzip**: Comprime archivos, reduciendo su tamaño. Ejemplo:
   - `gzip archivo.txt`: Comprime el archivo, creando `archivo.txt.gz`.
4. **zgrep**: Permite utilizar **grep** en archivos comprimidos con **gzip**. Ejemplo:
   - `zgrep "palabra" archivo.gz`: Busca "palabra" dentro del archivo comprimido.
5. **wc**: Muestra el conteo de líneas, palabras y caracteres de un archivo. Ejemplo:
   - `wc archivo.txt`: Muestra el número de líneas, palabras y caracteres.
   - Parámetros comunes:
     - -l: Contar solo líneas.
     - -w: Contar solo palabras.

8. 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
ps > PRUEBA
chmod 710 prueba
chown r o o t : r o o t PRUEBA
chmod 777 PRUEBA
chmod 700 / e t c / passwd
passwd r o o t
rm PRUEBA
man / e t c / shadow
f i n d / –name * . c o n f
usermod r o o t –d /home/ newroot –L
cd / r o o t
rm *
cd / e t c
cp * /home –R
shutdown

8.

- `ls -l > prueba`
  - Acción: Lista los archivos en el directorio actual con formato largo y redirige la salida a un archivo llamado `prueba`.
  - Ejecución: Este comando funcionará sin problemas porque no requiere permisos especiales.
- `ps > PRUEBA`
  - Acción: Muestra los procesos en ejecución en el sistema y redirige la salida a un archivo llamado `PRUEBA`.
  - Ejecución: Funcionará correctamente porque no requiere permisos especiales para ver los procesos del usuario.
- `chmod 710 prueba`
  - Acción: Cambia los permisos del archivo `prueba` a `rwx--x---` (el propietario tiene todos los permisos, el grupo tiene permisos de ejecución, y otros no tienen acceso).

- Ejecución: Funcionará, ya que puedes cambiar los permisos de archivos que te pertenecen.

- `chown root:root PRUEBA`
  - Acción: Intenta cambiar el propietario y el grupo del archivo `PRUEBA` a `root`.
  - Ejecución: No se puede ejecutar porque no tienes permisos de root para cambiar la propiedad de un archivo. Se mostrará un mensaje de error.

- `chmod 777 PRUEBA`
  - Acción: Cambia los permisos de `PRUEBA` a `rwxrwxrwx` (todos tienen acceso de lectura, escritura y ejecución).
  - Ejecución: Funcionará siempre que el archivo te pertenezca, ya que puedes cambiar sus permisos.

- `chmod 700 /etc/passwd`
  - Acción: Intenta cambiar los permisos del archivo `/etc/passwd` a `rwx------` (solo el propietario tiene acceso completo).
  - Ejecución: No se puede ejecutar, ya que no tienes permisos de root para modificar los permisos en `/etc`.

- `passwd root`
  - Acción: Intenta cambiar la contraseña del usuario `root`.
  - Ejecución: No se puede ejecutar porque no tienes permisos de root. Solo el superusuario puede cambiar la contraseña del usuario `root`.

- `rm PRUEBA`
  - Acción: Elimina el archivo `PRUEBA`.
  - Ejecución: Funcionará si el archivo te pertenece y tienes permisos de escritura en el directorio.

- `man /etc/shadow`
  - Acción: Intenta abrir la página de manual del archivo `/etc/shadow`, que no tiene una página de manual asociada.
  - Ejecución: Mostrará un error porque `/etc/shadow` no es un comando con una página de manual. El archivo contiene las contraseñas encriptadas

del sistema.

- `find / -name *.conf`
  - Acción: Busca archivos que terminen en `.conf` en todo el sistema.
  - Ejecución: Puede ejecutarse, pero debido a la falta de permisos, mostrará muchos errores "Permission denied" al intentar acceder a directorios a los que el usuario no tiene acceso.
- `usermod root -d /home/newroot -L`
  - Acción: Intenta cambiar el directorio de inicio de `root` a `/home/newroot` y bloquear la cuenta.
  - Ejecución: No se puede ejecutar porque solo el superusuario puede modificar cuentas de usuario, especialmente la cuenta `root`.
- `cd /root`
  - Acción: Intenta cambiar al directorio `/root`, que es el directorio de inicio de `root`.
  - Ejecución: No se puede ejecutar porque no tienes permisos para acceder al directorio `/root`.
- `rm *`
  - Acción: Elimina todos los archivos en el directorio actual.
  - Ejecución: Funcionará si tienes permisos de escritura en el directorio y eres el propietario de los archivos.
- `cd /etc`
  - Acción: Cambia el directorio a `/etc`.
  - Ejecución: Funcionará si tienes permisos de lectura en ese directorio, lo cual es común para usuarios regulares.
- `cp * /home -R`
  - Acción: Copia recursivamente todos los archivos y directorios de `/etc` al directorio `/home`.
  - Ejecución: No se puede ejecutar por completo debido a la falta de permisos para copiar algunos archivos en `/etc` (muchos de los cuales requieren privilegios de root).
- `shutdown`

- Acción: Apaga el sistema.
- Ejecución: No se puede ejecutar porque el usuario regular no tiene permisos para apagar el sistema, solo root puede hacerlo.

9. Indique qué comando sería necesario ejecutar para realizar cada una de las siguientes acciones:
   (a) Terminar el proceso con PID 23.
   (b) Terminar el proceso llamado init o systemd. ¿Qué resultados obtuvo?
   (c) Buscar todos los archivos de usuarios en los que su nombre contiene la cadena “.conf”
   (d) Guardar una lista de procesos en ejecución el archivo /home/<su nombre de usuario>/procesos
   (e) Cambiar los permisos del archivo /home/<su nombre de usuario>/xxxx a:
   Usuario: Lectura, escritura, ejecución
   Grupo: Lectura, ejecución
   Otros: ejecución
   (f) Cambiar los permisos del archivo /home/<su nombre de usuario>/yyyy a:
   Usuario: Lectura, escritura.
   Grupo: Lectura, ejecución
   Otros: Ninguno
   (g) Borrar todos los archivos del directorio /tmp
   (h) Cambiar el propietario del archivo /opt/isodata al usuario iso2010
   (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.

a. kill 23
b. killall init → killall systemd
c. find /home -name “* .conf”
d. ps aux > /home/<su nombre de usuario>/procesos
e. chmod u=rwx,g=rx,o=x /home/<su nombre de usuario>/xxxx

f. chmod u=rw,g=rx,o= /home/<su nombre de usuario>/xxxx

g. rm -rf /tmp/*

h. chown iso2010 /opt/isodata

i. pwd >> /home/<su nombre de usuario>/donde

11. 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.
(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".
(c) Cambie los permisos del archivo LEAME, de manera que se puedan ver reflejados los
siguientes permisos:
Dueño: ningún permiso
Grupo: permiso de ejecución
Otros: todos los permisos
(d) Vaya al directorio /etc y verifique su contenido. Cree un archivo dentro de su directorio
personal cuyo nombre sea leame donde el contenido del mismo sea el listado de todos
los archivos y directorios contenidos en /etc. ¿Cuál es la razón por la cuál puede crear
este archivo si ya existe un archivo llamado "LEAME.en este directorio?.
(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.
(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".

a. mkdir 23087-2 → cd 23087-2

b. vim LEAME → i → *INSERTAR DATOS* → *ESC* → :qw!

c. chmod u=,g=x,o=rwx LEAME

d. cd /etc → ls -l > /home/<su_nombre_de_usuario>/leame

▼ ¿Cuál es la razón por la cuál puede crear
este archivo si ya existe un archivo llamado "LEAME en este directorio?.

Razón por la cual se puede crear este archivo aunque exista uno llamado "LEAME": Los nombres de archivos en sistemas Linux son **sensibles a mayúsculas y minúsculas**, por lo que "LEAME" y "leame" son considerados archivos diferentes.

e. find / -name "nombre_del_archivo”

   a. Para buscar varios archivos con características similares, puedes usar comodines (`*`) o criterios avanzados. Por ejemplo, para buscar todos los archivos `.txt`: find / -name "*.txt”

   b. **Concepto teórico**: `find` recorre el árbol de directorios de forma recursiva y permite especificar una variedad de criterios de búsqueda (nombre, tipo, tamaño, permisos, etc.).

f. find / -name “.so” > /home/23087-2/.ejerciciof

12. 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
cd . / iso ; ps > f0
ls > f1
cd /
echo $HOME
ls –l $> $HOME/ iso / ls

cd $HOME; mkdir f2
ls –ld f2
chmod 341 f2
touch dir
cd f2
cd ~/ iso
pwd > f3
ps | g rep 'ps' | wc –l >> . . / f 2 / f 3
chmod 700 . . /f2; cd . .
find . –name etc/passwd
find / –name etc/passwd
mkdir ejercicio5

- `mkdir iso`:

  Crea un directorio llamado `iso` en el directorio de trabajo actual.

  Acción: Se crea el directorio `iso`.

  Ejecutable: Sí.

- `cd ./iso ; ps > f0`:
  - `cd ./iso`: Cambia el directorio de trabajo actual al directorio `iso`.
  - `ps > f0`: Ejecuta el comando `ps` (que lista los procesos) y redirige la salida al archivo `f0`.

    Acción: Cambia al directorio `iso` y guarda la lista de procesos en el archivo `f0`.**Ejecutable**: Sí.

- `ls > f1`:

  Lista el contenido del directorio actual y redirige la salida al archivo `f1`.

  Acción: Crea el archivo `f1` con el contenido del listado del directorio.

  Ejecutable: Sí.

- `cd /`:

  Cambia el directorio de trabajo al directorio raíz `/`.

  Acción: Se cambia al directorio raíz.

  Ejecutable: Sí.

- `echo $HOME`:

  Muestra el valor de la variable de entorno `$HOME`, que contiene el directorio personal del usuario actual.

  Acción: Imprime el valor del directorio personal del usuario.

  Ejecutable: Sí.

- `ls -l $> $HOME/iso/ls`:
  - `ls -l`: Lista el contenido del directorio actual con detalles.
  - `$> $HOME/iso/ls`: Aquí hay varios errores:
    - `ls -l $>` no es una sintaxis válida.
    - Se espera que se utilice `>` para redirigir la salida, pero el símbolo `$>` no tiene significado.

      Acción: Este comando no se puede ejecutar debido a errores de sintaxis.

    - **Ejecutable**: No.

- `cd $HOME; mkdir f2`:
  - `cd $HOME`: Cambia el directorio de trabajo al directorio personal del usuario.
  - `mkdir f2`: Crea un directorio llamado `f2` en el directorio personal.
  - **Acción**: Se cambia al directorio personal y se crea el directorio `f2`.
  - **Ejecutable**: Sí.

- `ls -ld f2`:

  Lista los detalles del directorio `f2`, sin mostrar su contenido, solo la información del directorio en sí.

  Acción: Muestra información sobre el directorio `f2`.

  Ejecutable: Sí.

- `chmod 341 f2`:

  Cambia los permisos del directorio `f2` a `341`, que significa:
  - **Propietario**: Permisos de ejecución (3 es equivalente a `-x`).
  - **Grupo**: Permiso de lectura (4 es equivalente a `r--`).

- **Otros**: Permiso de escritura (1 es equivalente a `-w`).

  Acción: Cambian los permisos del directorio `f2`.

- **Ejecutable**: Sí.

- `touch dir`:

  Crea un archivo vacío llamado `dir` en el directorio actual, o actualiza la marca de tiempo si el archivo ya existe.

  Acción: Crea o actualiza el archivo `dir`.

  Ejecutable: Sí.

- `cd f2`:

  Cambia el directorio de trabajo al directorio `f2`.

  Acción: Cambia al directorio `f2`.

  Ejecutable: Sí.

- `cd ~/iso`:

  Cambia el directorio de trabajo al directorio `iso` que se encuentra en el directorio personal del usuario.

  Acción: Cambia al directorio `iso` dentro del directorio personal.

  Ejecutable: Sí.

- `pwd > f3`:

  Muestra la ruta completa del directorio actual y redirige esa salida al archivo `f3`.

  Acción: Guarda la ruta del directorio actual en el archivo `f3`.

  Ejecutable: Sí.

- `ps | grep 'ps' | wc -l >> ../f2/f3`:

  - `ps`: Lista los procesos actuales.
  - `grep 'ps'`: Filtra las líneas que contienen el término `ps`.
  - `wc -l`: Cuenta las líneas de la salida anterior.
  - `>> ../f2/f3`: Añade el resultado al archivo `f3` ubicado en el directorio `f2`.

Acción: Añade el número de procesos que contienen "ps" en su nombre al archivo `f3`.

- **Ejecutable**: Sí.

- `chmod 700 ../f2; cd ..`:
  - `chmod 700 ../f2`: Cambia los permisos del directorio `f2` a `700` (solo el propietario tiene todos los permisos).
  - `cd ..`: Cambia al directorio padre del directorio actual.
  - **Acción**: Cambian los permisos de `f2` y luego se mueve al directorio padre.
  - **Ejecutable**: Sí.

- `find . -name etc/passwd`:

  Busca en el directorio actual y sus subdirectorios un archivo o directorio llamado `etc/passwd`.

  Acción: Busca el archivo `passwd` dentro de `etc` en el directorio actual.

  Ejecutable: Sí, aunque es probable que no encuentre ningún resultado.

- `find / -name etc/passwd`:

  Busca en todo el sistema de archivos un archivo o directorio llamado `etc/passwd`.

  Acción: Busca el archivo `passwd` dentro de `etc` en todo el sistema.

  Ejecutable: No completamente, porque no se puede acceder a algunos directorios sin permisos de root, lo que generará errores de permiso.

- `mkdir ejercicio5`:

  Crea un directorio llamado `ejercicio5` en el directorio actual.

  Acción: Crea el directorio `ejercicio5`.

  Ejecutable: Sí.

13. 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" directorio de trabajo /home/usuario.
- Copie el archivo "f4" en el directorio "dir11".
- Haga los mismo que en el inciso anterior pero el archivo de destino, se debe llamar
  "f7".
- Cree el directorio copia dentro del directorio usuario y copie en él, el contenido de
  "dir1".
- Renombre el archivo "f1"por el nombre archivo y vea los permisos del mismo.
- 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
- Renombre los archivos "f3 2 "f4"de manera que se llamen "f3.exe 2 "f4.exerespectivamente.
- 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

1. mkdir -p dir1/dir11
   mkdir -p dir2/dir21touch f1 f2 f3 f4
   touch dir1/f5
   touch dir2/dir21/f6
2. mv dir2/dir21/f3 /home/usuario/

3. cp f4 dir1/dir11/
4. cp f4 dir1/dir11/f7
5. mkdir /home/usuario/copia
   cp -r dir1/* /home/usuario/copia/
6. mv f1 archivo
   ls -l archivo
7. chmod 764 archivo
8. mv f3 f3.exe
   mv f4 f4.exe
9. chmod 026 f3.exe f4.exe

14. 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.
(b) Copie todo el contenido del directorio /var/log en el directorio creado en el punto
anterior.
(c) Empaquete el directorio creado en 1, el archivo resultante se debe llamar "misLogs.tar".
(d) Empaquete y comprima el directorio creado en 1, el archivo resultante se debe llamar
"misLogs.tar.gz".
(e) Copie los archivos creados en 3 y 4 al directorio de trabajo de su usuario.
(f) Elimine el directorio creado en 1, logs.
(g) Desempaquete los archivos creados en 3 y 4 en 2 directorios diferentes.

a. mkdir /tmp/logs
b. cp -r /var/log/* /tmp/logs/
c. tar -cvf misLogs.tar -C /tmp logs
d. tar -czvf misLogs.tar.gz -C /tmp logs

e. cp misLogs.tar misLogs.tar.gz ~/

f. rm -rf /tmp/logs

g. mkdir ~/misLogs1
   tar -xvf ~/misLogs.tar -C ~/misLogs1

   mkdir ~/misLogs2
   tar -xzvf ~/misLogs.tar.gz -C ~/misLogs2