Práctica 2

1. **Editor de textos**
   a. **Nombre al menos 3 editores de texto que puede utilizar desde la línea de comandos.**
      - Nano: editor de texto simple el cual es útil para editar archivos de configuración del sistema.
      - Vi/Vim: editor de texto que ofrece características y funciones que tiene modos de operación como inserción y comando.
      - Emacs: editor personalizado que tiene modos específicos para diferentes lenguajes de programación.

   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.**

      Un editor de texto es un programa usado para crear y modificar archivos de texto, mientras que los comandos cat, more y less son instrucciones para ver el contenido de archivos de texto sin la necesidad de modificarlos.

      En cuanto a los modos de operación de VI, son los siguientes

      - Modo de comando: modo predeterminado de VI que se usa para realizar acciones como mover el cursor, buscar texto, copiar y pegar, entre otras.
      - Modo de inserción: se puede escribir y editar el contenido del archivo.
      - Modo de visualización: usado para ver el contenido del archivo estáticamente.
      - Modo de selección: permite seleccionar texto para su posterior edición o eliminación.
      - Modo de ex: usado para realizar operaciones avanzadas, como guardar cambios, buscar y reemplazar texto, entre otras.

   c. **Nombre los comandos más comunes que se le pueden enviar al editor de textos VI.**

      Modo de comando:

      - i: entra al modo de inserción.
      - x: elimina el carácter bajo el cursor.
      - yy: copia la línea actual.
      - p: pega el contenido copiado en el portapapeles.
      - dd: elimina la línea actual.
      - u: deshace la última operación.
      - :w: guarda los cambios realizados en el archivo.
      - :q: sale del editor.
      - :q!: sale del editor sin guardar los cambios.
      - :wq: guarda los cambios realizados y sale del editor.

      Modo de inserción:

      - Esc: sale del modo de inserción y vuelve al modo de comando.

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.**
      i. La BIOS realiza una serie de pruebas y comprobaciones del hardware como RAM, CPU, almacenamiento, etc.
      ii. Se busca y carga el boot loader.
      iii. El cargador de arranque carga el kernel y le pasa el control.
      iv. Una vez cargado el kernel, inicializa y configura los dispositivos de hardware como procesador, memoria, controladores, etc.
      v. EL SO inicia los proceso del espacio del usuario como servicios del sistema, demonio de inicio de sesión, etc.

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

      El proceso INIT lo ejecuta el kernel al arrancar el sistema. Su objetivo es iniciar, mantener los servicios y procesos necesarios para que el SO funcione y detener los procesos en cada nivel de ejecución.

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

      El comando pstree muestra una representación de los procesos en ejecución en el SO. Se observa una estructura de árbol que muestra la relación de los procesos.

      Cada proceso está en una línea individual que tiene su PID y nombre de proceso. Los procesos del sistema se nombran en mayúscula y los de los usuarios en minúscula.

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

Los runlevels clasifican el estado del SO en diferentes modos de funcionamiento, cada uno con servicios y procesos específico.

Su objetivo es controlar los servicios y procesos que se ejecutan según la función en la situación en la que se encuentren.

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?**

- Runlevel 0: apagado del sistema.
- Runlevel 1: modo de usuario único o de mantenimiento del sistema, sin servicios de red ni servicios de múltiples usuarios.
- Runlevel 2: modo de múltiples usuarios sin servicios de red.
- Runlevel 3: modo de múltiples usuarios con servicios de red y sin entorno gráfico.
- Runlevel 4: no se usa de forma predeterminada, los usuarios pueden personalizarlo.
- Runlevel 5: modo de múltiples usuarios con servicios de red y con entorno gráfico.
- Runlevel 6: reinicio del sistema.

El runlevel por defecto que se ejecuta al iniciar el SO se define en /etc/inittab.

f. **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?**

Su finalidad es iniciar el sistema y cambiar diferentes niveles de ejecución. Almacena información sobre los procesos que deben iniciarse en cada nivel de ejecución y otros parámetros de configuración relacionados con la inicialización del sistema.

En su organización, cada línea contiene información sobre un proceso que debe ocurrir durante el arranque del sistema. Donde

- `id`: es un identificador único para la línea.
- `runlevel`: es el nivel de ejecución en el que se debe iniciar el proceso o realizar la acción. Puede ser un número del 0 al 6, o las letras `S` o `s` para el modo de usuario único.
- `action`: especifica la acción que se debe realizar. Las acciones comunes incluyen `initdefault` para establecer el nivel de ejecución predeterminado, `sysinit` para procesos de inicialización del sistema, `respawn` para reiniciar un proceso si termina inesperadamente, y `once` para ejecutar un proceso solo una vez.
- `process`: es el nombre del proceso o comando que se debe ejecutar.

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 de runlevel se puede usar el comando init seguido por el número de runlevel deseado.

El cambio no es permanente ya que el SO volverá al runlevel predeterminado después de un reinicio.

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

Los scripts RC son archivos de script utilizados para configurar y arrancar servicios y aplicaciones durante el proceso de inicio del sistema en sistemas Unix y Linux. Su finalidad principal es automatizar este proceso y asegurar que los servicios y aplicaciones se inicien en el orden correcto y con la configuración adecuada.

Se almacenan en el directorio /etc/init.d.

Cuando un sistema GNU/Linux arranca o se detiene, el programa "init" es el encargado de gestionar los procesos y servicios. Para determinar qué script ejecutar en cada acción, "init" utiliza el archivo de configuración "inittab", en el cual se especifica el nivel de ejecución (runlevel) del sistema y los scripts que deben ejecutarse para cada nivel de ejecución. Los scripts RC tienen un nombre que comienza con una letra "S" o "K", seguida de un número y el nombre del script, y se ejecutan en un orden específico dependiendo de su nivel de ejecución y del número que le acompaña.

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

Insserv es un comando que se usa para configurar los scripts de inicio del sistema y su orden de ejecución. Garantiza que los scripts de inicio se ejecuten en el orden correcto.

La ventaja de insserv respecto a un arranque tradicional incluyen una gestión más eficiente y organizada de los scripts de inicio, administración más fácil y control sobre los servicios que se inician en el arranque del sistema.

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

Upstart es un sistema de inicialización de servicios en sistemas GNU/Linux que utiliza una arquitectura basada en eventos y tareas para gestionar el proceso de arranque del sistema. Los eventos desencadenan la ejecución de tareas o procesos definidos en archivos de configuración, y pueden generar eventos adicionales que desencadenan otras tareas. Upstart permite definir dependencias entre los servicios y tareas para controlar el orden de inicio y evitar conflictos entre los servicios durante el arranque del sistema.

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

	Arquitectura	Inicio paralelo	Depuración	Registro de eventos	Compatibilidad
SystemV	Usa una arquitectura basada en scripts que se ejecutan secuencialmente durante el proceso de arranque.	Inicia los servicios secuencialmente.		SystemV no registra eventos y se basa en scripts para ejecutar tareas específicas.	SystemV es más compatible con las distribuciones de Linux antiguas y puede ser

					más fácil de mantener y administrar en algunos casos.
Upstart	Usa una arquitectura basada en eventos que permite mayor flexibilidad y control sobre el proceso de arranque.	Permite la inicialización paralela de servicios, lo que puede acelerar el proceso de arranque del sistema.	Proporciona herramientas de depuración y monitoreo más avanzado que SystemV, facilitando la identificación y solución de problemas en el proceso de arranque.	Registra los eventos y los utiliza para desencadenar acciones.	Upstart se ha diseñado para ser más moderno y más adecuado para entornos con mayores requisitos de rendimiento y flexibilidad.

l. **¿Qué reemplaza a los scripts RE de SystemV en Upstart? ¿En qué ubicación del filesystem se encuentran?**

En upstart los scripts RC reemplazan a los scripts RX de SystemV. Estos archivos RX definen los servicios que se deben iniciar o detener durante el proceso de arranque o apagado del sistema.

Los archivos RC de Upstart se encuentran en el directorio "/etc/init" del sistema de archivos. Cada archivo RC de Upstart tiene un nombre que refleja el nombre del servicio que define, seguido de la extensión ".conf". Por ejemplo, el archivo RC para el servicio "ssh" se llamaría "ssh.conf".

m. **Dado el siguiente job de upstart perteneciente al servicio de base de datos MySql indique a qué hace referencia cada línea del mismo.**

```
# MySQL Service
description 'MySQL Server' #describe el servicio que define
author 'info autor' #indica autor del mismo
start on (net-device-up #indica las condiciones de cuando debe inicar
  and local-filesystems
  and runlevel [2345]
stop on runlevel [016] #indica las conficiones de cuando debe detenerse
[...]
exee /usr/sbin/mysqld #indica la ruta y nombre del archivo que se debe ejecutar
[...]
```

n. **¿Qué es sytemd?**

Systemd es un sistema init monolitico usado como proceso padre para iniciar y detener los servicios del SO. Es una alternativa de SystemV y Upstart.

o. **¿A qué hace referencia el concepto de activación de socket en systemd?**

El concepto de activación de socket en systemd se refiere a una característica que permite que los servicios se activen automáticamente cuando se recibe una solicitud de conexión en un socket determinado.

Un socket es un canal de comunicación bidireccional que se utiliza para intercambiar datos entre procesos.

p. **¿A qué hace referencia el concepto de cgroup?**

Cgroup es una funcionalidad que permite limitar, controlar y aislar los recursos del sistema utilizados por los procesos y grupos de procesos.

3. **Usuarios**

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

La información se guarda en distintos archivos

- /etc/passwd: contiene información básica de los usuarios, como su nombre de usuario, ID de usuario (UID), ID de grupo principal (GID), nombre completo, directorio de inicio y shell predeterminada.
- /etc/shadow: información de contraseñas encriptadas para los usuarios. Solo es accesible por el usuario root y se usa para proteger la información de contraseñas de los usuarios.
- /etc/group: contiene información sobre los grupos de usuarios, como su nombre y ID de grupo.
- /etc/gshadow: contiene información de contraseñas encriptadas para los grupos de usuarios. Solo accesible por el usuario root y se usa para proteger la información de contraseñas de los grupos de usuarios.

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

- UID (User ID): número único que se le asigna a un usuario para identificarlo de manera única en el sistema.
- GUID (Group ID): número para identificar a cada grupo del sistema.

No pueden coexistir dos UIDs iguales.

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 es el usuario administrador del sistema GNU/Linux. Tendrá permisos de acceso y control totales sobre el sistema y puede hacer cualquier acción en el mismo.

En Linux solo puede haber un usuario con este perfil y su UID es 0.

d. **Agregue un nuevo usuario llamado iso2017 a su instalación de GNU/Linux, especifique que su home sea creada en /home/iso_2017, y hágalo miembro del grupo catedra (si no existe, deberá crearlo). Luego, sin iniciar sesión como este usuario**

cree un archivo en su home personal que le pertenezca. Luego de todo esto, borre el usuario y verifique que no queden registros de él en los archivos de información de los usuarios y grupos.

```
sudo groupadd catedra #crea el gurpo catedra
sudo useradd iso2017 -d /home/iso_2017 -m  iso_2017  #crea el usuario iso2017, indica el directorio y lo crea, indica el usuario al que corresponde
sudo usermod -a -G catedra iso2017 #lo agrega al grupo cátedra
sudo -u iso2017 touch /home/iso2017/archivo.txt #crea archivo sin iniciar sesión con iso2017
userdel -r iso2017  # elimina usuario y sus directorios y archivosInvestigue la funcionalidad y parámetros de los siguientes comandos:
```

- **useradd** ó **adduser**: usado para crear un nuevo usuario en el sistema.

Parámetro	Función
-c	Usado para agregar un comentario al usuario.
-d	Especifica el directorio home del usuario.
-e	Establece la fecha de vencimiento de la cuenta del usuario.
-g	Especifica el grupo principal del usuario.
-m	Crea automáticamente el directorio home del usuario si aún no existe. En caso que no se use este parámetro, deberá crearse manualmente.
-s	Especifica el shell de inicio del usuario.
-u	Especifica el identificador del usuario (UID).

- **usermod**: modifica atributos de un usuario existente

Parámetro	Función
-c	Agrega o modifica un comentario.
-d	Modifica el directorio home del usuario.
-e	Establece la fecha de vencimiento para la cuenta.
-g	Modifica el grupo principal del usuario.
-l	Modifica nombre de usuario.
-s	Modifica shell de inicio del usuario.
-u	Modifica UID

- **userdel**: elimina un usuario del sistema

Parámetro	Función
-f	Elimina el usuario incluso si aún tiene procesos o archivos abiertos.
-r	Elimina también el directorio home y los archivos asociados.

- **su**: cambia de usuario a una cuenta de root. Permite tener acceso a privilegios de super usuario

Parámetro	Función
-	Inicia sesión de shell como si el usuario de destino hubiera iniciado sesión
-c	Usado para ejecutar un comando en la cuenta de destino sin iniciar sesión de shell completa.
-s	Especifica qué shell se utilizará después de iniciar la sesión de super usuario.
-u	Cambia a un usuario sin iniciar sesión.

- **groupadd**: crea un nuevo grupo de usuarios en el sistema

Parámetro	Función
-g	Especifica el GID para el nuevo grupo.
-r	Crea un grupo del sistema en lugar de un grupo normal. Los grupos del sistema se usan para procesos o servicios del sistema.
-f	Fuerza la creación del grupo incluso si ya existe con ese mismo nombre

- **who**: muestra la lista de usuarios que iniciaron sesión en el sistema.

Parámetro	Función
-a	Muestra información adicional como la hora de inicio e IP.
-b	Muestra fecha y hora que se inició la última vez.
-q	Muestra el número de usuarios en lugar de la lista.
-s	Muestra nombre de usuario y hora que inició.
-u	Muestra información sobre tiempo de inactividad.

- **groupdel**: elimina grupos existentes en el sistema.

Parámetro	Función

-f	Fuerza la eliminación aunque tenga usuarios.

- **passwd**: permite a los usuarios cambiar su contraseña o la de otro usuario si tienen el permiso. Solo lo puede usar sudo.

Parámetro	Función
-d	Elimina la contraseña.
-l	Bloquea la cuenta del usuario.
-u	Desbloquea la cuenta del usuario

4. **FileSystem**
   a. **¿Cómo son definidos los permisos sobre archivos en un sistema GNU/Linux?**

      Los permisos de acceso se dividen en 3 categorías: propietario, grupo al que pertenece el archivo y los demás usuarios además de establecer 3 tipos de accesos: lectura, escritura, ejecución (R,W,X).

      R=4, W=2, X=1.

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

      i. **chmod**: cambia los permisos de acceso de un archivo y directorio

      ii. **chown**: cambia el propietario o grupo propietario de un archivo/ directorio.

Parámetro	Función
u	Especifica el nuevo propietario del archivo
g	Especifica el nuevo propietario del archivo
R	Modifica de manera recursiva el propietario o grupo de un archivo y todos el contenido

      iii. **chgrp**: cambia el grupo propietario de un archivo o directorio. El grupo propietario es el grupo al que pertenece el archivo y determina los permisos de quienes no pertenecen a ese grupo.

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

      La notación octal representa los permisos de acceso a un archivo usando base 8. Cada permiso se representa de un número de 0 a 7.

      - 0: sin permisos (---)
      - 1: permiso de ejecución (--x)
      - 2: permiso de escritura (-w-)
      - 3: permiso de escritura y ejecución (-wx)
      - 4: permiso de lectura (r--)
      - 5: permiso de lectura y ejecución (r-x)
      - 6: permiso de lectura y escritura (rw-)
      - 7: permiso de lectura, escritura y ejecución (rwx)

   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.**

      En teoría no debería pasar.

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

      Linux tiene dos formas de especificar la ubicación de un archivo.

      - Full path name: ruta completa que especifica la ubicación desde el directorio raíz del sistema. Comienza con /.
      - Relative path name: ruta que especifica la ubicación en relación con el directorio actual.

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

      Para determinar directorio actual se usa el comando pwd.

      Puedo acceder a mi directorio actual sin escribir todo el path usando ~. Este comando lo llevará al directorio personal del usuario actual. Si deseo acceder al directorio personal de otro usuario, puede usar el tilde seguido del nombre del usuario.

   g. **Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el uso de FileSystem**

      i. **cd**: usado para cambiar el directorio actual.

Parámetro	Función
cd..	Cambia el directorio actual al directorio padre.
cd -	Cambia al directorio anterior al último directorio en el que estuvo.
cd /	Cambia al directorio raíz del sistema.

ii. **umount**: desmonta sistemas de archivos. Desmontar se refiere al proceso de eliminar la conexión entre el sistema de archivos y el SO.

Parámetro	Función
-f	Fuerza el desmontaje.
-l	Realiza el desmontaje en un momento en el que no se esté usando el sistema.
-n	Simula lo que ocurriría si se desmonta el sistema.
-t	Especifica tipo de sistema de archivos que se desea desmontar.
-v	Muestra información sobre el proceso de desmontaje.
-r	Realiza un desmontaje recursivo.

iii. **mkdir**: crea un nuevo directorio en el sistema de archivos.

Parámetro	Función
-p	Crea directorios padres si no existen.
-m	Establece los permisos del directorio que se creará.
-v	Muestra un mensaje detallado para cada directorio creado.

iv. **du**: muestra el espacio en disco de los archivos y directorios en el sistema

Parámetro	Función
-a	Muestra el tamaño de todos los archivos, incluyendo los ocultos.
-h	Muestra los archivos en formado KB, MB, GB, etc.
-s	Muestra solo el tamaño total de los especificados.
-c	Muestra el tamaño total de los archivos junto con el tamaño individual.
-x	Muestra el uso de espacio sin incluir otros sistemas de archivos.

v. **rmdir**: elimina directorios vacíos.

Parámetro	Función
-p	Elimina el directorio especificado junto con cualquier directorio vacío que lo contenga.
-v	Muestra información sobre el proceso.

vi. **df**: muestra información del espacio disponible en los sistemas de archivos montados.

vii. **mount**: usado para montar sistemas de archivos. Para ejecutarlo se debe tener permisos root.

Parámetro	Función
-t	Especifica el tipo de sistema que se montará.
-o	Especifica las opciones que se usarán en el sistema.
-n	Realiza una simulación del montaje.
-v	Muestra información detallada del proceso.
-r	Monta el sistema en modo de solo lectura.

viii. **ln**: crea enlaces entre archivos. Un enlace duro es una asociación directa entre dos nombres de archivos y los simbólicos es un tipo especial de archivo que actúa como puntero a otro archivo.

Parámetro	Función
-s	Crea un enlace simbólico en lugar de un enlace duro.
-f	Sobrescribe cualquier enlace existente.
-v	Muestra información sobre el proceso.

ix. **ls**: lista el contenido de un directorio.

Parámetro	Función
-l	Lista en formato largo, con información de permisos, propietario, tamaño, fecha de modificación, etc.
-a	Muestra todos los archivos, incluso los ocultos.
-h	Muestra el tamaño de cada uno en tamaño KB, MB, GB.
-t	Ordena los archivos por fecha y hora.
-r	Ordena los archivos descendentemente.
-d	Muestra solo el nombre del directorio

x. **pwd**: imprime en pantalla el directorio actual.

xi. **cp**: copia archivos o directorios.

Parámetro	Función
-r	Copia directorios recursivamente.
-i	Pregunta si se desea sobrescribir un archivo existente.
-v	Muestra información sobre el proceso de copia.
-p	Conserva los atributos originales del archivo, como permisos, propietario, etc.
-u	Copia solo los archivos no existentes en el destino.
-n	No sobrescribe archivos existentes.

xii. **mv**: mueve o renombra archivos y directorios.

Parámetro	Función
-i	Pregunta si desea sobrescribir archivo existente.
-u	Mueve solo archivos no existentes en destino.
-v	Muestra información sobre el proceso.
-f	Fuerza la acción sin preguntar si desea sobrescribir
-n	No sobrescribe existentes.

5. Procesos

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

Es una tarea que está en control del kernel. Cada proceso tiene un PID (Process Identifier) para identificarlo. Además cada proceso tiene un PPID (Parent Process Identifier) que infica el PID del proceso padre que lo creó y es útil para comprender la jerarquía de procesos.

Todos los procesos tiene PID y PPID asignados porque serán necesarios para su ejecución. Tiene otros atributos como

- Estado del proceso: indica si está en ejecución, suspendido o detenido.
- Uso de CPU: muestra el porcentaje de uso de CPU del proceso.
- Prioridad del proceso: indica la prioridad del proceso en relación al resto.
- Memoria usada: cantidad de memoria que usa el sistema.
- Tiempo de ejecución: tiempo de CPU usado por el proceso hasta el momento.

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

- **ps**: muestra la lista de procesos en ejecución pero solo los pertenecientes al usuario actual.
  - -e: muestra todos los procesos en ejecución.
  - -f: muestra información detallada sobre cada proceso, incluyendo propietario, PIC, estado, consumo de recursos, etc.
  - -u <usuario>: procesos pertenecientes al usuario especificado.

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

- Background: está ejecutándose en segundo plano y no interactúa directamente con el usuario.
- Foreground: se ejecuta en primer plano e interactúa mediante la terminal o sea que se espera una respuesta del usuario mientras está ejecutándose.

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

Para ejecutar un proceso en background, podemos agregar el caracter "&" al final del comando que estamos ejecutando en la terminal.

Para el paso de back a foreground puedo usar el comando fg.

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

Se usa para conectar dos o más comandos y redirigir la salida del primero como entrada del primero, creando una cadena de procesos. Permite que la salida de un comando sea la entrada de otro.

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

Hay 3 tipos de redirecciones

- **Redirección de entrada:** toma la entrada de un archivo en lugar del teclado, Usa el símbolo < para especificar el archivo de entrad. Ejemplo: se enviará el contenido del archivo “entrada.txt: como entrada para el comando grep

```
grep "palabra" < entrada.txt
```

- **Redirección de salida estándar**: envía la salida de un comando a un archivo en lugar de la pantalla. Se usa el símbolo > para especificar el archivo de salida. Ejemplo, enviará la salida del comando ls al archivo salida.txt.

```
ls > salida.txt
```

- **Redirección de errores estándar**: enviará los errores de un comando a un archivo en lugar de la pantalla. Se usa el símbolo 2> para especificar el archivo de errores. Ejemplo, enviará los errores de ls a “errores.txt”

```
ls no_existe 2> errores.txt
```

&> redirige salida estándar y salida de error.

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

El comando kill envía una señal a un proceso para interrumpir su ejecución. La señal que se envía es SIGTERM que indica que el proceso debe finalizar de forma controlada.

El comando puede recibir como parámetro el PID.

```
kill 1234 #finaliza el proceso con PID 1234
kill 1234 5678 #finaliza los procesos con PID 1234 y 5678
```

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

i. **ps**: muestra la información de procesos en ejecución.

Parámetro	Función
aux	Muestra una lista detallad de todos los procesos, incluyendo los de otros usuarios.
-e	Muestra todos los procesos en ejecución.
-f	Muestra una lista detallada incluyendo información de padre, tiempo de ejecución y uso de CPU.
-u <usuario>	Muestra los procesos ejecutándose del usuario especificado.
-e | grep <proceso>	Busca un proceso específico en la lista de procesos en ejecución.
-p <PID>	Muestra información del proceso con el PID indicado.

ii. **kill**: ídem

iii. **pstree**: ídem

iv. **killall**: envía señales de terminación a procesos que coinciden con el nombre especificado. Killall trabaja con los nombres del proceso.

Parámetro	Función
-e	Envia señales solo a procesos que sean exactamente igual al nombre del proceso indicado.
-i	Pide una confirmación antes de enviar la señal.
-u	Envía la señal solo a procesos pertenecientes al usuario indicado.

v. **top**: monitorea procesos ejecutándose en tiempo real.

Parámetro	Función
-u	Muestra los procesos del usuario especificado.
-p	Muestra la información de los procesos especificados por el PID.
-o	Ordena la salida de top según un criterio.
-b	Genera la salida en modo batch.
-d	Especifica el intervalo de actualización de top.

vi. **nice**: establece la prioridad de ejecución de un proceso. La prioridad determina cuánto tiempo de CPU se le asignará y qué tan rápido se ejecutará.

La sintaxis es

```
nice [opciones] [comando[argumentos]]
```

6. Otros comandos de Linux

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

Empaquetar hace referencia a la acción de crear un archivo que contenga uno o varios archivos o directorios comprimidos en un solo archivo.

Hay varios formatos como tar, zip, gzip.

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?**

```
du -ch archivo1 archivo2 archivo3 archivo4 #suma tamaños
tar -czvf archivos.tar.gz archivo1 archivo2 archivo3 archivo4 #empaqueta
ls -lh archivo1 archivo2 archivo3 archivo4 archivos.tar.gz #compara tamaños
```

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

```
tar -czvf archivo.tar.gz archivo1 archivo2 archivo3 archivo4

#-c indica que se creará un nuevo archivo,
#-z indica que se utilizará compresión gzip
#-v indica que se mostrará información detallada del proceso en la terminal
#-f indica que se especificará el nombre del archivo a crear
```

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

Ídem inciso c)

e. **Investigue la funcionalidad de los siguientes comandos:**

i. **tar**: crea, manipula y desempaqueta archivos empaquetados.

Parámetro	Función
-c	Crea un nuevo archivo tar.
-x	Extrae el contenido de un archivo tar.
-f	Permite especificar el nombre del archivo tar que se va a crear o manipular.
-v	Muestra la lista de archivos que se están procesando.
-z	Comprime el archivo tar usando gzip.
-j	Comprime el archivo tar usando bzip2.
-C	Cambia el directorio antes de realizar cualquier función.

ii. **grep**: busca patrones de texto en uno o varios archivos.

Parámetro	Función
-i	Realiza búsqueda de forma insensible a mayúsculas y minúsculas.
-v	Muestra todas las líneas que no coinciden con el patrón de búsqueda.
-n	Muestra el número de línea en la que se encuentra el patrón de búsqueda.
-r	Realiza una búsqueda recursiva en todos los archivos o subdirectorios.
-w	Busca el patrón de búsqueda como una palabra completa y no como una cadena de caracteres dentro de una palabra larga.
-c	Muestra el número de líneas que coinciden con el patrón de texto.

iii. **gzip**: comprime archivos en formato GNU. Al comprimir un archivo con gzip, se crea un archivo con extensión .gz.

Parámetro	Función
-c	Envía la salida comprimida a la salida estándar.
-d	Descomprime el archivo comprimido, eliminando el original
-r	Comprime todos los archivos en un directorio y sus subdirectorios de forma recursiva.
-t	Verifica la integridad.
-v	Muestra información sobre el proceso.
-f	Fuerza la compresión incluso si ya existe.

iv. **zgrep**: es similar a grep pero está diseñado para buscar patrones en archivos comprimidos con gzip. La funcionalidad es la misma que grep pero en vez de leer un archivo de texto plano, puede leer un archivo comprimido con gzip y buscar un patrón en el contenido del archivo.

Parámetro	Función
-i	Busca patrones sin importar mayúsculas o minúsculas.
-v	Muestra las líneas que no coinciden con el patrón de búsqueda.
-c	Muestra el número de líneas que coinciden con el patrón de búsqueda.

-l	Muestra el nombre de los archivos que contienen el patrón de búsqueda.
-h	No muestra el nombre del archivo en la salida.
-r	Busca de manera recursiva en todos los archivos dentro.

v. **wc**: cuenta el número de líneas, palabras y caracteres de un archivo o entrada de texto.

Parámetro	Función
-l	Cuenta solo el número de líneas.
-w	Cuenta solo el número de palabras.
-c	Cuenta solo el número de bytes o caracteres.
-m	Cuenta el número de caracteres, incluyendo caracteres multibyte,
-L	Muestra la longitud de la línea más larga.

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 de forma detallada y lo redirecciona al archivo prueba
ps > PRUEBA  #almacena el estado de los procesos en el archivo PRUEBA
chmod 710 prueba #cambia los permisos del archivo prueba, el propietario puede RWX, grupo solo X y otros no pueden hacer nada
chown root : root PRUEBA # cambia el propietario de PRUEBA a root y el grupo también
chmod 777 PRUEBA #cambia los permisos de PRUEBA, todos podrán RWX
chmod 700 /etc/passwd #cambia los permisos del archivo passwd, propietario podra RWX y el resto nada
passwd root #cambia la contraseña de usario root
rm PRUEBA #elimina el archivo PRUEBA
man /etc/shadow #IDK
find / −name * . conf #busca en / los archivos que tengan extensión .conf
usermod root −d /home/newroot −L #modifica el directorio del usuario root y lo bloquea
cd / root #cambia al directorio raiz del usuario root
rm * #elimina todo lo del directorio actual
cd /etc # cambia al directorio /etc
cp * /home −R #copia todo lo del directorio /home recursivamente
shutdown #apaga el sistema
```

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?**

```
killall init
#aunque no es recomendado porque es escencial para que el sistema funcione
```

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

```
find /home -name "*\.conf"
```

El comando `grep` se utiliza para buscar patrones específicos dentro de archivos de texto y mostrar las líneas que contienen esos patrones. Por ejemplo, se puede utilizar para buscar todas las líneas en un archivo de registro que contienen una cadena de búsqueda específica.

Por otro lado, el comando `find` se utiliza para buscar archivos y directorios que cumplen ciertos criterios, como nombre de archivo, tamaño, fecha de modificación, etc.

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

```
ps aux > /home/iso2017/procesos
```

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

i. Usuarios: Lectura, escritura, ejecución.

ii. Grupo: Lectura, ejecución.

iii. Otros: ejecución.

```
chmod 750 /home/iso2017/xxxx
#R=4, W=2, X=1.
```

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

i. Usuario: Lectura, escritura.

ii. Grupo: Lectura, ejecución.

iii. Otros: ninguno.

```
chmod 650 /home/iso2017/yyyy
```

g. **Borrar todos los archivos del directorio /tmp.**

```
rm /tmp*
```

h. **Cambiar el propietario del archivo /opt/isodata al usuario iso2021.**

```
chown iso2021 /opt/isodata
```

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

```
pwd >> /home/iso2017/donde
```

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

a. **Ingrese al sistema como usuario “root”.**

```
su root.
```

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 savila
sudo passwd savila
```

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

```
find / -mtime 0 #busca archivos modificados en las ultimas 24 horas
ls -lt / | grep "d"
```

d. **Crear un directorio en /tmp llamado cursada2021**

```
mkdir /tmp/cursada2021
```

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

```
cp /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.**

```
sudo chown -R savila:users /tpmp/micursada2021/
```

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

```
chmod -R 764 /*
```

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.

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

```
ps -e | wc -l #ps -e muestra la cantidad de comandos y wc -l cuenta la cant de lineas
```

k. **Verifique la cantidad de usuarios conectados al sistema.**

```
who -q
```

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

```
write savila "hola"
```

m. **Apague el sistema.**

```
shutdown
```

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

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

```
mkdir 185443
cd 185443
```

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 sin ningún permiso, grupo con permiso de ejecución y otros con 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 -a > /home/leame
```

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

```
find[ruta] -name [nombre_del_archivo]
```

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" > /home/185443/.ejercicio_f #el asterisco indica que es una cadena al final
```

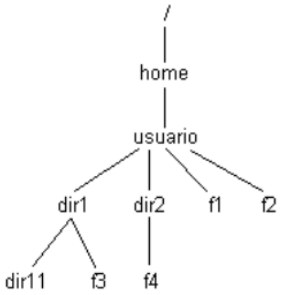
11. I**ndique qué acción realiza cada uno de los comandos indicados a continuación considerando su orden. Suponga que se ejecutan desde un usuario que no es root ni pertenece al grupo de root. (Asuma que se encuentra posicionado en el directorio de trabajo del usuario con el que se logueó). En caso de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

```
mkdir iso #crea la carpeta iso
cd . /iso #vuelve al path /iso
ps > f0  #f0 guarda los procesos ejecutándose
ls > f1 #guarda en f1 el listado del contenido del directorio actual
cd / #vuelve al directorio raíz
echo $HOME #muestra el path completo del directorio raíz
ls −l & > $HOME/iso/ls #redirige al path %HOME/iso/ls el listado detallado del contenido del directorio
cd $HOME #se posiciona en el directorio raíz
mkdir f2 #crea la carpeta f2
ls −ld f2 #muestra el contenido del directorio f2 detalladamente la información del directorio
chmod 341 f2 #cambia los permisos de f2
touch dir #imprime dir ??
cd f2 #se ubica en f2
cd ~/iso #cambia de directorio a /iso
pwd > f3 #guarda en f3 el direcotrio que nos encontramos
ps | grep ' ps ' | wc −l >> . . /f2/f3 #en f2/f3 guarda las lineas que coinciden con el patron de grep ps
chmod 700 . . /f2 #cambia los permisos de /f2
cd . . #cambia al directorio padre
find . −name etc/passwd #busca en el directorio actual todos los archivos con nombre etc/passwd
find / −name etc/ # busca en la raíz los archivos que tengan nombre etc/
```

```
mkdir ejercicio5 #crea carpeta ejercicio5
............
............
```

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

```
cd /home/sofia
mkdir dir1
mkdir dir2
mkdir dir1/dir11
touch dir1/f3
touch dir2/f4
touch f1
touch f2
```



a. **Utilizando la estructura de directorios anteriormente creada, indique que comandos son necesarios para realizar las siguientes acciones:**
    a. **Mueva el archivo f3 al directorio de trabajo /home/usuario.**

    ```
    mv /home/usuario/dir1/f3 /home/usuario/
    ```

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

    ```
    cp /home/usuario/dir2/f4 /home/usuario/dir1/dir11/
    ```

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

    ```
    cp -i /home/usuario/dir2/f4 /home/usuario/dir1/f7/
    ```

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

    ```
    mkdir /home/usuario/copia
    cp -r /home/usuario/dir1/* /home/usuario/copia/
    ```

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

    ```
    mv /home/usuario/f1 /home/usuario/archivo ##indica la ruta y tmb el archivo
    ls -l /home/usuario/archivo #muestra permisos del archivo archivo
    ```

    f. **Cambie los permisos del archivo llamado archivo de manera de reflejar lo siguiente:**
        a. Usuario: permisos de lectura y escritura.
        b. Grupo: permisos de ejecución.
        c. Otros: todos los permisos.

    ```
    chmod u+rw,g+x,o+rwx /home/usuario/archivo
    ```

    g. **Renombre los archivos "f3 "f4"de manera que se llamen "f3.exe 2 "f4.exe respectivamente.**

    ```
    mv /home/usuario/dir1/f3 /home/usuario/dir1/f3.exe
    mv /home/usuario/dir2/f3 /home/usuario/dir2/f4.exe
    ```

h. **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 u+,g+r,o+wx /home/usuario/dir1/f3.exe /home/usuario/dir2/f4.exe
```

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

a. **Cree un directorio llamado logs en el directorio /tmp.**

```
mkdir /tmp/logs/
```

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

```
cp -r /tmp/logs/* /var/log #/* indica que tiene que copiar todo el contenido
```

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

```
tar -cf  misLogs.tar /tmp/logs ##primero indico el nombre y dsp destino
```

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

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

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

```
cp misLogs.tar.gz misLogs.tar /home/usuario
```

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 -xvf misLogs.tar -C /home/usuario/n1
gzip -d misLogs.tar.gz -C /home/usuario/n2/
```