* 1)
  + **nano**: Un editor de texto fácil de usar que es ideal para principiantes. Proporciona atajos de teclado visibles en la parte inferior de la pantalla.

**vim**: Un editor de texto avanzado y altamente configurable. Es conocido por su eficiencia, pero tiene una curva de aprendizaje más pronunciada. Se basa en modos de operación, como el modo de inserción y el modo de comando.

**emacs**: Otro editor de texto potente y extensible, que también se puede utilizar como un entorno de desarrollo. Ofrece una amplia gama de funciones y es altamente personalizable.

* + La diferencia que presenta un Editor de Texto de los comandos “cat”, “more” o “less” es que son herramientas diferentes utilizadas para diferentes propósitos en el manejo de archivos de texto en sistemas Unix like. Principales diferencias:
* Editor de Texto: Aplicación que te permite crear, modificar y guardar archivos de texto de manera interactiva.
* “cat”: Comando que se utiliza para mostrar el contenido de uno o varios archivos de texto en la salida estándar de consola. Se puede usar para ver el contenido completo de un archivo.
* “more”: Es un visualizador de texto paginado. Muestra el contenido de un archivo de manera incremental, una página a la vez.
* “less”: Es similar al “more” ya que se utiliza para ver el contenido de un archivo de texto, ya que permite avanzar y retroceder a través del archivo. También permite buscar texto y realizar otras acciones de navegación dentro del archivo.
* **Modos de operación que posee el editor Vi:**  Modo de Comandos, de Inserción, de Reemplazo, de Visualización, de Línea de Comandos, de Bloque, de Inserción de Línea y de Comandos de Ex.
  + Modo de Comando (predeterminado al abrir un archivo):
* **h**: Mover el cursor a la izquierda.
* **j**: Mover el cursor hacia abajo.
* **k**: Mover el cursor hacia arriba.
* **l**: Mover el cursor a la derecha.
* **gg**: Ir al inicio del archivo.
* **G**: Ir al final del archivo.
* **$**: Ir al final de la línea actual.
* **0**: Ir al inicio de la línea actual.

Edición:

* **i**: Entrar en modo de inserción antes del cursor.
* **a**: Entrar en modo de inserción después del cursor.
* **o**: Abrir una nueva línea debajo y entrar en modo de inserción.
* **x**: Borrar el carácter bajo el cursor.
* **dd**: Eliminar la línea actual.
* **yy**: Copiar la línea actual.
* **p**: Pegar el contenido copiado o eliminado después del cursor.
* **u**: Deshacer el último cambio.
* **Ctrl + r**: Rehacer el último cambio deshecho.

Guardar y salir

* **:w**: Guardar (escribir) los cambios.
* **:q**: Salir de vi.
* **:wq**: Guardar y salir.
* **:q!**: Salir sin guardar cambios.

Búsqueda y reemplazo

* **/texto**: Buscar hacia adelante la cadena "texto".
* **?texto**: Buscar hacia atrás la cadena "texto".
* **n**: Ir a la siguiente ocurrencia de la búsqueda.
* **N**: Ir a la ocurrencia anterior de la búsqueda.
* **:s/texto1/texto2/g**: Reemplazar todas las ocurrencias de "texto1" por "texto2" en la línea actual.

Pasaje a los demás modos:

* Empezamos en Modo Comandos.
* “R”: Pasamos a Modo de Reemplazo.
* “:”: Pasamos a Modo de Línea de Comandos.
* “Ctrl + v”: Pasamos a Modo de Bloque.
* “O”: Inserta una línea encima de la línea actual.
* “o”: Inserta una línea debajo de la línea actual.
* “Esc”: Pasamos a Modo de Comandos.
* 2)
  + Pasos del proceso de Arranque:

1. Se empieza a ejecutar el código del BIOS.
2. El BIOS ejecuta el POST.
3. El BIOS lee el sector de arranque (MBR).
4. Se carga el gestor de arranque (MBC).
5. El bootloader carga el kernel y el initrd.
6. Se monta el initrd como file system raíz y se inicializan componentes esenciales.
7. El Kernel ejecuta el proceso init y se desmonta el initrd.
8. Se lee el /etc/inittab.
9. Se ejecutan los scripts del runlevel 1
10. El final del runlevel 1 le indica que vaya al runlevel por defecto.
11. Se ejecutan los scripts del runlevel por defecto.
12. El sistema está listo para usarse.
    * El proceso INIT tiene como objetivo llevar al sistema desde un estado de apagado o reinicio hasta un estado funcional completamente cargado y listo para que los usuarios y otros procesos puedan interactuar con él, cumpliendo con las siguientes funciones:

* Cargar todos los subprocesos necesarios para el correcto funcionamiento del Sistema Operativo.
* Es el encargado de montar los filesystems y de hacer disponible los demás dispositivos.

Este proceso es ejecutado por el Kernel en procesos de arranque que usan System V.

* + Los RunLevels son el modo en que arranca Linux, cada uno es responsable de levantar (iniciar) o bajar (parar) una serie de servicios. Su objetivo principal es definir el estado en el que un sistema operativo se encuentra durante su arranque y el conjunto de servicios y entornos que están disponibles en ese estado. Cada runlevel tiene un propósito específico y configura el sistema para un uso particular.
  + Runlevels según el estándar:
* **Runlevel 0** (halt): Este nivel se utiliza para apagar el sistema de manera segura. Todos los servicios se detienen y se apaga el hardware de manera adecuada.
* **Runlevel 1** (Single User Mode): Este nivel se utiliza para tareas de mantenimiento y recuperación del sistema. El sistema arranca en modo de usuario único con acceso mínimo.
* **Runlevel 2** (Multiuser without NFS): Se usa para iniciar el sistema en modo multiusuario sin habilitar la red. Es útil en situaciones de servidores locales.
* **Runlevel 3** (Full Multiuser Mode Console): Similar al Runlevel 2 pero con servicios de red habilitados.
* **Runlevel 4** (No estándar): Este nivel no se utiliza en muchas distribuciones y por lo tanto no tiene una configuración estándar.
* **Runlevel 5** (X11 Modo Gráfico o Multiusuario con Interfaz Gráfica): Este nivel incluye todos los servicios de Runlevel 3, también inicia un sistema de escritorio o entorno gráfico.
* **Runlevel 6** (Reboot): En este nivel, el sistema se reinicia. La configuración del runlevel al iniciar un sistema operativo Unix like se encuentra definida en “/etc/inittab”.

No todas las distribuciones respetan estos estándares clásicos de **System V Init**, ya que muchas adoptan **systemd** u otros sistemas de inicio modernos.

* + El archivo **/etc/inittab** es un archivo de configuración utilizado por el sistema de inicialización **System V (SysV Init)**. Su principal finalidad es definir cómo se comporta el proceso de arranque del sistema, qué runlevel debe ejecutarse por defecto y qué acciones deben realizarse en cada uno de esos runlevels.

Contiene líneas de configuración que definen las acciones a realizar en diferentes situaciones en diferentes runlevels, y cómo manejar procesos específicos.

Estructura de la información: id:runlevels:action:process

**id**: Un identificador único de 1-4 caracteres para la entrada.

**runlevels**: Lista de runlevels en los que se debe ejecutar el proceso (por ejemplo, 2345).

**action**: La acción que se debe realizar.

**process**: El comando o script que se ejecuta cuando se activa la entrada.

* + → init Y

→ telinit Y

Comandos que se utilizan para cambiar el runlevel del sistema. Normalmente requieren permisos de 9 superusuario por eso se utiliza junto con “sudo”. En este caso “Y” debería cambiarse por el número del runlevel.

El cambio se aplica de forma **temporal** hasta que el sistema se reinicie o se apague. Al reiniciar, el sistema volverá al **runlevel por defecto** configurado en el archivo **/etc/inittab**

* + Los **scripts RC (Run Command)** son scripts de inicialización que se ejecutan durante el arranque o apagado de un sistema GNU/Linux. Su finalidad es inicializar, configurar y detener servicios y procesos según el **runlevel** o estado del sistema. Cada vez que el sistema cambia de **runlevel**, los scripts RC ejecutan comandos para iniciar o detener servicios, dependiendo del nuevo runlevel en el que el sistema se encuentra. Se almacenan en los directorios **/etc/rcX.d/**, donde **X** es el runlevel (por ejemplo, **/etc/rc3.d/** para el runlevel 3): Contiene enlaces simbólicos que apuntan a los scripts en **/etc/init.d/**.

Cuando el sistema cambia de runlevel (por ejemplo, cuando se inicia o se apaga), el **proceso init** revisa el runlevel y ejecuta los scripts adecuados para ese nivel.

**Runlevel específico**: El sistema busca en el directorio correspondiente al nuevo runlevel, por ejemplo **/etc/rc3.d/** para el runlevel 3.

**Scripts con nombres especiales**: En los directorios **/etc/rcX.d/**, los scripts tienen nombres con prefijos especiales que indican si deben iniciarse o detenerse:

* **S:** Los scripts que empiezan con S (de Start) se ejecutan para iniciar servicios.
* **K:** Los scripts que empiezan con K (de Kill) se ejecutan para detener servicios.

El **orden de ejecución** de los scripts se determina por los **números** en sus nombres. Los scripts se ejecutan en orden numérico ascendente.

[S|K][Orden][Nombre Script]

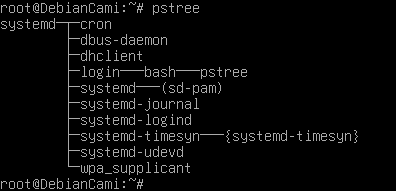
* 3)
  + **systemd** es un sistema de inicio (init system) y conjunto de herramientas de administración de sistemas para Linux. Fue diseñado para mejorar la eficiencia del arranque, la gestión de procesos y la administración general del sistema, reemplazando el sistema tradicional de inicio **SysV Init** en muchas distribuciones modernas de Linux. systemd permite la ejecución de servicios en paralelo, lo que acelera el proceso de arranque del sistema.

En lugar de **runlevels**, systemd utiliza **targets**. Los targets son un conjunto de units que se ejecutan en conjunto para alcanzar un estado determinado. Por ejemplo, multi-user.target es equivalente al runlevel 3 y graphical.target es equivalente al runlevel 5 (modo gráfico).

* + En **systemd**, una **Unit** es el concepto central que representa y gestiona un recurso o servicio del sistema. Cada **Unit** es un archivo de configuración que describe cómo controlar y gestionar un recurso específico en el sistema, como un servicio, un dispositivo de hardware, un punto de montaje, o incluso un conjunto de dependencias entre servicios.

Existen diferentes tipos de **Units:**

* **Service Units (.service)**: Representan servicios del sistema que deben ejecutarse, detenerse o reiniciarse.
* **Target Units (.target)**: Agrupan varios servicios y otros tipos de units. Los targets son equivalentes a los **runlevels** en los sistemas antiguos basados en SysV Init.
* **Socket Units (.socket)**: Representan sockets de comunicación, generalmente de red o de interprocesos. Estos sockets activan automáticamente un servicio cuando hay actividad.
* etc.
  + El comando **systemctl** en **systemd** es la herramienta principal para interactuar con el sistema de inicialización **systemd**. Se utiliza para gestionar y controlar los **services**, **targets** y otras **units** del sistema. Permite iniciar, detener, habilitar, deshabilitar, reiniciar y ver el estado de los servicios y otros recursos gestionados por **systemd**.
  + En **systemd**, un **target** es una agrupación lógica de **units** que permite coordinar el arranque y la organización de servicios y otros recursos del sistema. Los **targets** son equivalentes a los antiguos **runlevels** en los sistemas basados en **SysV Init**. Los **targets** se utilizan para llevar el sistema a un estado específico o conjunto de condiciones en las que ciertos servicios y recursos están disponibles. No ejecutan comandos directamente, sino que definen un punto en el proceso de arranque o apagado donde determinadas **units** deben estar activas o inactivas.
  + pstree → Lo que se ve es la estructura jerárquica de los procesos en el sistema de forma gráfica. Muestra una representación visual del árbol de procesos.



* 4)
  + En un sistema GNU/Linux, la información de los usuarios se almacena principalmente en varios archivos de configuración ubicados en el directorio **/etc**. Estos archivos contienen los detalles de las cuentas de usuario, contraseñas, grupos y privilegios.

**/etc/passwd**: Contiene la información básica de todas las cuentas de usuario. Cada línea representa un usuario y tiene un formato de campos separados por dos puntos (**:**).

**/etc/shadow**: Contiene las contraseñas encriptadas de los usuarios y otra información relacionada con la seguridad de las cuentas.

Solo los usuarios con privilegios (como root) pueden leer este archivo, ya que contiene información sensible.

**/etc/group**: Define los grupos de usuarios del sistema.

**/etc/gshadow**: Almacena la información de contraseñas encriptadas para grupos, similar a **/etc/shadow** pero para grupos.

* + Las siglas **UID** hacen referencia a **User Identifier** o **Identificador de Usuario**. Es un número único que se asigna a cada usuario en un sistema GNU/Linux para identificarlo de forma unívoca. El **UID** es utilizado por el sistema para gestionar los permisos de acceso a archivos y recursos.

Las siglas **GID** hacen referencia a **Group Identifier** o **Identificador de Grupo**. Es un número único que identifica a un grupo en el sistema. Los grupos son utilizados para asignar permisos colectivos a varios usuarios.

GNU/Linux está diseñado para que cada usuario tenga un **UID único**. Esto asegura que el sistema pueda gestionar de manera efectiva los permisos y derechos de acceso. Compartir **UIDs** rompe esta lógica y puede causar serios problemas de seguridad y administración, por lo que se debe evitar.

* + **Usuario Root**: Es el administrador del sistema o superusuario. Tiene privilegios administrativos completos y puede realizar cualquier operación en el sistema. Este usuario es esencial para la administración y configuración del sistema, puede detener el sistema, instalar software en el mismo, modificar o reconfigurar el kernel, drivers, etc.

Solo puede existir un usuario con ese perfil en GNU/Linux. Su UID y GID es 0 (cero).

* + → groupadd catedra

→ useradd -m -d /home/iso\_2017 -g catedra iso2017

* **-m**: Crea el directorio **home** si no existe.
* **-d /home/iso\_2017**: Especifica el directorio **home** del usuario.
* **-g catedra**: Hace que el usuario sea miembro del grupo **catedra**.

→ touch /home/iso\_2017/archivo.txt (**touch**: Crea un archivo vacío llamado **archivo.txt** en el directorio **home** del usuario)

→ chown iso2017:catedra /home/iso\_2017/archivo.txt (**chown iso2017:catedra**: Cambia el propietario del archivo al usuario **iso2017** y el grupo **catedra**)

→ ls -l /home/iso\_2017 (verificamos que el archivo haya sido creado correctamente y que pertenezca al usuario **iso2017**)

→ userdel -r iso2017 (elimina el usuario **iso2017** con el comando **userdel**, **-r**: Elimina el directorio **home** del usuario y su contenido.)

→ cat /etc/passwd | grep iso2017 (Verifica que el usuario **iso2017** ya no exista en el sistema revisando el archivo **/etc/passwd**)

→ cat /etc/group | grep catedra (verificar que no hay registros en **/etc/group**)

* + El comando **useradd/adduser** es una herramienta básica y directa que crea un nuevo usuario en el sistema.
* useradd [Opciones] nombre\_usuario
* **-m**: Crea el directorio home del usuario si no existe.
* **-d /ruta/home**: Especifica una ruta personalizada para el directorio home.
* **-g**: Especifica el grupo principal al que pertenece el usuario.
* **-G**: Define grupos adicionales a los que pertenecerá el usuario.
* **-s /bin/bash**: Establece el intérprete de comandos (shell) del usuario. Ejemplo: /bin/bash.
* **-c**: Agrega un comentario o descripción del usuario.
* **-e**: Especifica la fecha de expiración de la cuenta en formato AAAA-MM-DD.
* **-f**: Define el número de días después de la expiración de la contraseña para desactivar la cuenta.
* **-p**: Establece la contraseña del usuario encriptada.
* **-r**: Crea una cuenta de sistema (sin un directorio home ni capacidad de iniciar sesión).

El comando **usermod** se utiliza para modificar las propiedades y configuración de los usuarios existentes en un sistema GNU/Linux. Con este comando, puedes cambiar el grupo principal o secundario de un usuario, actualizar su directorio home, cambiar su shell de inicio, y más.

* usermod [Opciones] nombre\_usuario
* **-l nuevo\_login**: Cambia el nombre de usuario.
* **-d /nueva\_ruta\_home**: Cambia el directorio **home** del usuario.
* **-g grupo**: Cambia el grupo primario del usuario.
* **-G grupo1,grupo2,...**: Añade o cambia los **grupos secundarios** a los que pertenece el usuario.
* **-aG**: Añade el usuario a un grupo **sin eliminarlo de otros grupos secundarios**.
* **-s /ruta/shell**: Cambia la shell predeterminada del usuario.
* **-L** y **-U**: **-L**: Bloquea la cuenta del usuario (deshabilita el inicio de sesión). **-U**: Desbloquea la cuenta.
* **-e AAAA-MM-DD**: Establece una fecha de expiración para la cuenta del usuario.
* **-f**: Especifica el número de días tras la expiración de la contraseña antes de desactivar la cuenta. Un valor de **-1** desactiva esta funcionalidad.

El comando **userdel** se utiliza para eliminar cuentas de usuario en un sistema GNU/Linux. Cuando eliminas un usuario, puedes optar por conservar o eliminar el directorio **home** y los archivos asociados a ese usuario.

* userdel [Opciones] nombre\_usuario
* **-r**: Elimina el directorio **home** del usuario y sus archivos asociados.
* **-f**: Fuerza la eliminación del usuario incluso si está actualmente logueado o si tiene procesos en ejecución.
* **--remove-all-files**: Borra todos los archivos propiedad del usuario en el sistema, sin importar dónde estén ubicados.

El comando **su** (abreviatura de "substitute user" o "switch user") permite cambiar de usuario en un sistema GNU/Linux sin necesidad de cerrar la sesión actual. El uso más común de este comando es para cambiar al usuario **root** (el superusuario) y obtener privilegios administrativos, pero también se puede usar para cambiar a otros usuarios.

* su [Opciones] nombre\_usuario
* **-** (guión): Inicia un nuevo entorno de sesión como si el usuario se hubiera logueado directamente, cargando las variables de entorno y el perfil del usuario (por ejemplo, /etc/profile, .bashrc).
* **-c**: Ejecuta un comando específico como el usuario designado, sin necesidad de iniciar una nueva sesión interactiva.
* **-s /ruta/shell**: Inicia sesión con una shell específica en lugar de la shell predeterminada del usuario.
* **-p** o **--preserve-environment**: Mantiene el entorno actual en lugar de cargar el entorno del usuario al que cambias.

El comando **groupadd** se utiliza para crear nuevos grupos en un sistema GNU/Linux. Los grupos son útiles para gestionar permisos y facilitar la administración de usuarios con permisos similares o comunes.

* groupadd [Opciones] nombre\_grupo
* **-g GID**: Especifica manualmente el **ID del grupo (GID)** que se asignará al grupo. Si no se especifica, el sistema asigna automáticamente el siguiente GID disponible.
* **-r**: Crea un **grupo del sistema**. Los grupos del sistema suelen tener GIDs más bajos y son usados por servicios y procesos del sistema en lugar de usuarios comunes.
* **-f**: Si el grupo ya existe, este parámetro no genera un error, simplemente no realiza ninguna acción. Si además se especifica un **GID**, y el grupo ya existe pero con un GID diferente, **groupadd** cambiará el GID del grupo existente.

El comando **who** se utiliza para mostrar información sobre los usuarios que están actualmente conectados al sistema. Este comando proporciona detalles como el nombre del usuario, el terminal (TTY) que están utilizando, la fecha y hora de inicio de sesión, y la dirección IP o hostname desde el cual se conectaron.

* who [Opciones]
* **-b**: Muestra la fecha y hora del último reinicio del sistema.
* **-q**: Muestra solo los nombres de los usuarios conectados y el número total de usuarios.
* **-H**: Muestra encabezados (columnas) en la salida.
* **-T**: Muestra el estado del terminal (TTY), indicando si los usuarios pueden recibir mensajes con **write**.
* **-u**: Muestra información adicional sobre el tiempo de inactividad (idle time) del usuario.
* **--ips**: Muestra solo las direcciones IP de los usuarios conectados (si están disponibles).

El comando **groupdel** se utiliza para eliminar grupos en un sistema GNU/Linux.

* groupdel nombre\_grupo
* **-r**: Elimina el grupo y el directorio asociado al mismo en /home si ha de existir.

El comando **passwd** se utiliza para cambiar la contraseña de un usuario en un sistema GNU/Linux. Este comando puede ser utilizado por los usuarios para cambiar sus propias contraseñas, y por el superusuario (root) para cambiar las contraseñas de otros usuarios.

* passwd [Opciones] nombre\_usuario
* **-d**: Elimina la contraseña del usuario, lo que significa que el usuario podrá acceder sin necesidad de contraseña.
* **-e**: Fuerza a que el usuario cambie su contraseña en el próximo inicio de sesión.
* **-l**: Bloquea la cuenta del usuario, deshabilitando su acceso. Esto se logra añadiendo un símbolo **!** al inicio de la contraseña en el archivo **/etc/shadow**.
* **-u**: Desbloquea la cuenta del usuario, permitiendo el acceso nuevamente.
* **-S**: Muestra el estado de la cuenta y la contraseña del usuario.
* 5)
  + Permisos que se definen en archivos:



* **Lectura**: Permite al usuario leer el contenido de un archivo o directorio y listar su contenido con el comando **ls**.
* **Escritura**: Permite la escritura, modificación y eliminación, en caso de directorios, la escritura permite crear nuevos archivos o borrar archivos existentes.
* **Ejecución**: Permite a un usuario ejecutar el archivo si este
* es un programa, en caso de directorio, permite entrar al
* mismo usando el comando **cd**.

Los permisos se representan mediante una cadena de diez caracteres cuando se usa el comando ls -l.

La representación típica es:

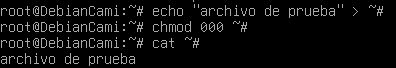
* **-rwxr-xr--**
* **El primer carácter** indica el tipo de archivo (-: archivo normal, d: directorio l: enlace simbólico)
* rwx: El propietario tiene permisos de lectura, escritura y ejecución.
* r-x: El grupo tiene permisos de lectura y ejecución, pero no de escritura.
* r--: Otros tienen solo permisos de lectura.
  + El comando **chmod** (change mode) se utiliza para cambiar los permisos de acceso de archivos y directorios en un sistema GNU/Linux.
* chmod [Opciones] modo archivo
* **-R**: Cambia los permisos de manera recursiva en todos los archivos y subdirectorios.
* **--reference=archivo**: Establece los permisos de un archivo igual que los de otro archivo existente.
* Los permisos se pueden establecer utilizando dos notaciones: **numérica** y **simbológica**.
* **Notación Numérica**:
  + Lectura (r) = 4
  + Escritura (w) = 2
  + Ejecución (x) = 1
  + Ej: chmod 754 archivo.txt [(7 = 4 + 2 + 1 = rwx) para el propietario, (5 = 4 + 1 = r-x) para el grupo, (4 = 4 = r--) para otros]
* **Notación Simbológica**:
  + **u**: propietario (user)
  + **g**: grupo (group)
  + **o**: otros (others)
  + **a**: todos (all)
  + **+**: agregar permisos
  + **-**: quitar permisos
  + **=**: establecer permisos exactos
  + Ej: chmod u+x archivo.txt (Agrega permiso de ejecución al propietario)

El comando **chown** (change owner) se utiliza para cambiar el propietario y, opcionalmente, el grupo de archivos y directorios en un sistema GNU/Linux.

* chown [Opciones] nuevo\_propietario[:nuevo\_grupo] archivo
* **nuevo\_propietario**: Especifica el nuevo propietario del archivo o directorio. Puede ser el nombre de usuario o el UID (ID de usuario).
* **nuevo\_grupo** (opcional): Especifica el nuevo grupo al que debe pertenecer el archivo. Puede ser el nombre del grupo o el GID (ID de grupo). Se separa del propietario por dos puntos (:). Si solo se desea cambiar el propietario, se omite el grupo.
* **-R**: Cambia de manera recursiva el propietario y el grupo para todos los archivos y subdirectorios dentro de un directorio.
* **--reference=archivo**: Cambia el propietario y el grupo del archivo especificado para que coincidan con los de otro archivo existente.

El comando **chgrp** (change group) se utiliza para cambiar el grupo asociado a un archivo o directorio en un sistema GNU/Linux.

* chgrp [Opciones] nuevo\_grupo archivo
* **nuevo\_grupo**: Especifica el nuevo grupo al que se desea cambiar el archivo. Puede ser el nombre del grupo o el GID (ID de grupo).
* **-R**: Cambia de manera recursiva el grupo para todos los archivos y subdirectorios dentro de un directorio.
* **--reference=archivo**: Cambia el grupo del archivo especificado para que coincida con el grupo de otro archivo existente.
  + RESPONDIDO EN LA PREGUNTA ANTERIOR. VER COMANDO “chmod” MODOS NUMÉRICOS.
  + Sí, existe la posibilidad de que algún usuario del sistema pueda acceder a determinado archivo para el cual no posee permisos, este sería el caso del usuario root ya que este último puede acceder y modificar cualquier archivo o directorio.

Ej: 

* + El **full path name** es la ruta completa que especifica la ubicación exacta de un archivo o directorio en el sistema de archivos, comenzando desde el directorio raíz (/). Esta ruta proporciona la jerarquía completa de directorios que se deben recorrer para llegar al archivo o directorio deseado.
* Ej: Si tenés un archivo llamado documento.txt ubicado en el directorio /home/usuario/Documentos, la ruta completa sería: /home/usuario/Documentos/documento.txt

El **relative path name** es una forma de especificar la ubicación de un archivo o directorio en relación con el directorio de trabajo actual. No comienza desde el directorio raíz, sino desde el directorio en el que te encuentras en ese momento.

* Ej para caso anterior: ./documento.txt
  + Se puede determinar en qué directorio te encuentras actualmente utilizando el comando **pwd**. Podes ingresar a tu directorio personal sin necesidad de escribir todo el path usando **cd** ó **cd ~**.
  + El comando **cd** (change directory) se utiliza para cambiar el directorio de trabajo actual en la línea de comandos.
* cd [Opciones] [Directorio]
* **directorio**: Especifica el directorio al que deseas cambiar. Puede ser una ruta absoluta (comenzando desde la raíz /) o una ruta relativa (en relación al directorio actual).
* **..**: Representa el directorio padre del directorio actual. Usar cd .. te llevará un nivel hacia arriba en la jerarquía de directorios.
* **.**: Representa el directorio actual. cd . no cambia el directorio, pero se puede usar en combinaciones.
* **~**: Representa el directorio personal del usuario actual. cd ~ te lleva directamente a tu home.
* **-**: Cambia al directorio anterior. Por ejemplo, si te encuentras en /home/usuario/Documentos y cambias a /tmp, puedes volver a Documentos con cd -.

El comando **umount** (unmount) se utiliza para desmontar sistemas de archivos que están actualmente montados en el sistema.

* umount [Opciones] [Punto de Montaje | Dispositivo]
* **punto\_de\_montaje**: Especifica la ubicación donde el sistema de archivos está montado. Por ejemplo, /mnt/usb.
* **dispositivo**: Puede especificarse el dispositivo a desmontar, como /dev/sdb1.
* **-a**: Desmonta todos los sistemas de archivos que están en el archivo /etc/mtab.
* **-f**: Fuerza el desmontaje del sistema de archivos, útil si el sistema de archivos está ocupado.
* **-l**: Desmonta el sistema de archivos de manera diferida (lazy unmount).

El comando **mkdir** (make directory) se utiliza para crear nuevos directorios en el sistema de archivos.

* mkdir [Opciones] directorio1 [directorio2 ...]
* **directorio**: El nombre del directorio que deseas crear. Puede ser una ruta absoluta o relativa.
* **-p**: Crea el directorio y, si es necesario, también crea los directorios padres.
* **-v**: Muestra un mensaje de confirmación por cada directorio que se crea.

El comando **du** (disk usage) se utiliza para estimar y mostrar el uso de espacio en disco de archivos y directorios. Es útil para identificar cuánta memoria ocupa cada directorio y sus contenidos.

* du [Opciones] [Ruta]
* **ruta**: Especifica el archivo o directorio cuyo uso de disco deseas verificar. Si no se proporciona, du mostrará el uso del directorio actual.
* **-h**: Muestra el tamaño en un formato legible para humanos (K, M, G).
* **-s**: Muestra solo el total del uso de disco para el directorio especificado, en lugar de listar cada subdirectorio.
* **-a**: Muestra el uso de disco para todos los archivos, no solo para los directorios.
* **-c**: Muestra un total acumulado después de listar el uso de todos los archivos y directorios.
* **-d <nivel>**: Limita la profundidad de los subdirectorios que se muestran. Por ejemplo, -d 1 mostrará solo el uso en el directorio actual y su primer nivel de subdirectorios.

El comando **rmdir** (remove directory) se utiliza para eliminar directorios vacíos en el sistema de archivos. Es importante notar que solo puede eliminar directorios que no contienen archivos o subdirectorios.

* rmdir [Opciones] directorio1 [directorio2 ...]
* **directorio**: Especifica el nombre del directorio que deseas eliminar. Puede ser una ruta absoluta o relativa.
* **-p**: Elimina el directorio especificado y sus directorios padres si también están vacíos.
* **--verbose o -v**: Muestra un mensaje por cada directorio que se elimina, confirmando la acción.

El comando **df** (disk free) se utiliza para mostrar información sobre el uso del espacio en disco de los sistemas de archivos montados. Es útil para verificar cuánto espacio libre hay disponible en las particiones y dispositivos de almacenamiento.

* df [Opciones] [Sistema de archivos]
* **sistema\_de\_archivos**: Especifica un sistema de archivos particular que deseas consultar. Si no se proporciona, df mostrará información sobre todos los sistemas de archivos montados.
* **-h**: Muestra la información en un formato legible para humanos (K, M, G) en lugar de en bloques de 1 KB.
* **-T**: Muestra el tipo de sistema de archivos junto con la información del uso.
* **-a**: Muestra todos los sistemas de archivos, incluidos los que tienen un uso de 0 bloques.
* **-i**: Muestra el uso de inodos en lugar del uso de espacio en disco.

El comando **mount** se utiliza para montar sistemas de archivos en un punto de montaje, permitiendo que el sistema operativo acceda a los archivos y directorios contenidos en esos sistemas.

* mount [Opciones] [Dispositivos] [Punto de Montaje]
* **dispositivo**: Especifica el dispositivo que deseas montar (por ejemplo, /dev/sdb1).
* **punto\_de\_montaje:** Especifica la ubicación en el sistema de archivos donde deseas montar el dispositivo (por ejemplo, /mnt/usb).
* **-t tipo:** Especifica el tipo de sistema de archivos (por ejemplo, ext4, ntfs, vfat). Si se omite, mount intentará detectarlo automáticamente.
* **-o opciones:** Permite especificar opciones adicionales como ro (solo lectura), rw (lectura/escritura), noexec (no permitir la ejecución de binarios), entre otras.
* **-a**: Monta todos los sistemas de archivos especificados en /etc/fstab que no estén ya montados.

El comando **ln** se utiliza para crear enlaces (links) en un sistema de archivos. Existen dos tipos principales de enlaces que puedes crear: **enlaces duros (hard links)** y **enlaces simbólicos (soft links o symlinks)**.

* ln [archivo\_original] [enlace\_duro]
* ln -s [archivo\_original] [enlace\_simbolico]
* **-s**: Crea un enlace simbólico en lugar de un enlace duro.
* **-f**: Fuerza la creación del enlace, sobrescribiendo cualquier archivo existente en la ubicación del enlace.
* **-v**: Muestra un mensaje detallado por cada enlace creado (modo verbose).
* **-i**: Solicita confirmación antes de sobrescribir un archivo existente.

El comando **ls** (list) se utiliza para listar los archivos y directorios dentro de un directorio específico.

* ls [Opciones] [ruta]
* **ruta**: Especifica la ruta del directorio que deseas listar. Si no se especifica, se listan los archivos del directorio actual.
* **-l**: Muestra la lista de archivos en formato largo, incluyendo detalles como permisos, número de enlaces, propietario, grupo, tamaño, y fecha de última modificación.
* **-a**: Muestra todos los archivos, incluidos los archivos ocultos (los que comienzan con un punto .).
* **-h**: Muestra los tamaños en un formato legible para humanos (por ejemplo, KB, MB, GB).
* **-R**: Lista los archivos de los subdirectorios de manera recursiva.
* **-t**: Ordena los archivos por fecha de modificación, mostrando primero los más recientes.
* **-S**: Ordena los archivos por tamaño, mostrando primero los más grandes.
* **-r**: Invierte el orden de la lista (por ejemplo, si se usa con -t, los archivos se ordenan del más antiguo al más reciente).
* **-d**: Muestra información solo sobre el directorio especificado, no su contenido.
* **-i**: Muestra el número de inodo para cada archivo.

El comando **pwd** (print working directory) muestra la ruta absoluta del directorio de trabajo actual en el que te encuentras dentro del sistema de archivos.

* pwd [Opciones]
* **-L**: Muestra la ruta lógica (la ruta con enlaces simbólicos, si existen).
* **-P**: Muestra la ruta física, resolviendo cualquier enlace simbólico en la ruta.

El comando **cp** (copy) se utiliza para copiar archivos y directorios de una ubicación a otra en un sistema de archivos.

* cp [Opciones] [Fuente] [Destino]
* **fuente**: Archivo o directorio que quieres copiar.
* **destino**: Lugar donde quieres copiar el archivo o directorio.
* **-r** o **-R**: Copia directorios de manera recursiva, incluyendo sus subdirectorios y contenido.
* **-i**: Interactivo, solicita confirmación antes de sobrescribir archivos.
* **-v**: Modo detallado, muestra los archivos y directorios que se están copiando (verbose).
* **-f**: Forza la copia y sobrescribe los archivos de destino sin preguntar.
* **-u**: Solo copia archivos si la fuente es más reciente que el archivo de destino o si el archivo de destino no existe.
* **-p**: Preserva los atributos del archivo original, como los permisos, tiempos de modificación, propietario y grupo.
* **-a**: Modo archivo. Equivalente a usar -dR --preserve=all. Copia recursivamente y conserva enlaces simbólicos, permisos, tiempos y otros atributos.

El comando **mv** (move) se utiliza para mover archivos y directorios de una ubicación a otra o para renombrar archivos o directorios.

* mv [Opciones] [Fuente] [Destino]
* **fuente**: Archivo o directorio que deseas mover o renombrar.
* **destino**: Nuevo nombre o ruta de destino.
* **-i**: Interactivo, solicita confirmación antes de sobrescribir archivos en el destino.
* **-f**: Fuerza el movimiento y sobrescribe los archivos de destino sin preguntar.
* **-n**: No sobrescribir archivos de destino si ya existen.
* **-v**: Modo detallado, muestra los archivos y directorios que se están moviendo o renombrando (verbose).
* 6)
  + **Proceso**: Es la instancia de ejecución de un programa de software, cada uno de estos procesos tiene su propio espacio de memoria y recursos asignados, y pueden estar en ejecución en paralelo con otros procesos. Tienen un program counter, un ciclo de vida y pasan por varios estados, además poseen un stack para ellos y uno para el kernel.
* **PID** (Process ID): Identificador único de un proceso en el sistema operativo, estos se asignan secuencialmente.
* **PPID** (Parent Process ID): Se refiere al PID del proceso padre que creó/inicio el proceso actual.

Todos los procesos tienen un PID y un PPID.

* + Comandos:
* **ps**: muestra una instantánea de los procesos en ejecución en ese momento.
* **top**: muestra una vista dinámica en tiempo real de los procesos del sistema, actualizando continuamente.
* **htop**: es una versión mejorada de top con una interfaz interactiva más amigable.
* **pstree**: muestra los procesos en forma de árbol, mostrando la jerarquía de los procesos en el sistema.
  + **Foreground** (Primer plano): Cuando un proceso se ejecuta en **Foreground**, significa que está **ocupando la terminal** desde la que fue iniciado. El usuario debe esperar a que el proceso finalice para poder ejecutar otros comandos en la misma terminal.

**Background** (Segundo plano): Cuando un proceso se ejecuta en **Background**, significa que se está ejecutando **sin ocupar la terminal**, permitiendo al usuario continuar ingresando otros comandos mientras el proceso sigue en ejecución.

* + Para ejecutar un proceso en segundo plano (**Background**) desde la línea de comandos, simplemente debes añadir un **&** al final del comando. Esto permite que el proceso se ejecute mientras la terminal queda libre para ejecutar otros comandos. Ej: ./mi\_programa &

Para pasar un proceso de foreground a background solo debemos de ejecutar el comando “bg %n” siendo n el PID y de background a foreground “fg %n” siendo n el PID.

* + El **pipe** (|) en sistemas operativos Unix y GNU/Linux es una herramienta muy poderosa que permite **redirigir la salida de un comando** como entrada de otro comando. Su finalidad es permitir la **combinación de múltiples comandos**, facilitando el procesamiento y filtrado de datos de manera eficiente.

Ej: ls -l | grep ".txt"

Este comando lista todos los archivos en el directorio actual con detalles (ls -l) y luego filtra los resultados para mostrar solo aquellos que tienen la extensión .txt.

Ej: ps aux | grep "python" | sort

Este comando lista todos los procesos en ejecución (ps aux), busca aquellos que contienen la palabra "python" (grep "python"), y luego ordena los resultados (sort).

* + Tipos de redirecciones:
* **Redirección de salida ( > ):** Redirige la salida estándar de un comando hacia un archivos, si el archivo existe lo sobreescribe, si no, lo crea. Comando > archivo
* **Redirección de salida añadiendo ( >> ):** Similar a > solo que no sobrescribe, añade la salida a la última línea del archivo. Comando >> archivo
* **Redirección de entrada ( < ):** Toma un archivo como entrada y lo pasa como entrada estándar para el comando. Comando < archivo
* **Redirección de línea de entrada ( << ):** Permite ingresar múltiples líneas de texto desde la terminal como entrada estándar de un comando. Comando << líneas\_de\_texto
  + El comando **kill** en sistemas GNU/Linux se utiliza para enviar señales a procesos en ejecución, permite la interacción con los procesos, como terminarlos de manera controlada o enviarles señales específicas. Requiere el uso del PID. Señales que envía:
* **SIGTERM (15)**: Solicita a un proceso que se termine de manera ordenada.
* **SIGKILL (9)**: Termina un proceso de manera inmediata y no puede ser ignorado.
* **SIGSTOP (19)**: Suspende un proceso.
* **SIGCONT (18)**: Reanuda un proceso que ha sido suspendido.

### **ps (Process Status):** Muestra una instantánea de los procesos en ejecución en ese momento. No actualiza la información en tiempo real como top.

* ps aux: Muestra todos los procesos del sistema (a=all, u=user-oriented output, x=processes without a controlling terminal).
* ps -ef: Muestra todos los procesos en un formato estándar (e=every process, f=full format).
* ps -p PID: Muestra información del proceso con el PID especificado.

**kill:** Envía señales a los procesos, generalmente para terminarlos.

* kill -9 PID: Fuerza la finalización del proceso con el PID especificado (SIGKILL).
* kill -15 PID: Solicita la terminación del proceso con el PID especificado (SIGTERM, por defecto).
* kill -l: Lista todas las señales que pueden ser enviadas.

**pstree**: Muestra los procesos en ejecución en forma de un árbol jerárquico, donde se puede ver la relación entre procesos padres e hijos.

* pstree -p: Muestra el PID de los procesos junto con su jerarquía.
* pstree -u: Muestra los usuarios propietarios de cada proceso.
* pstree -a: Muestra los argumentos de línea de comandos de los procesos.

**killall**: Envía una señal a todos los procesos que coincidan con un nombre determinado, en lugar de tener que utilizar PIDs específicos.

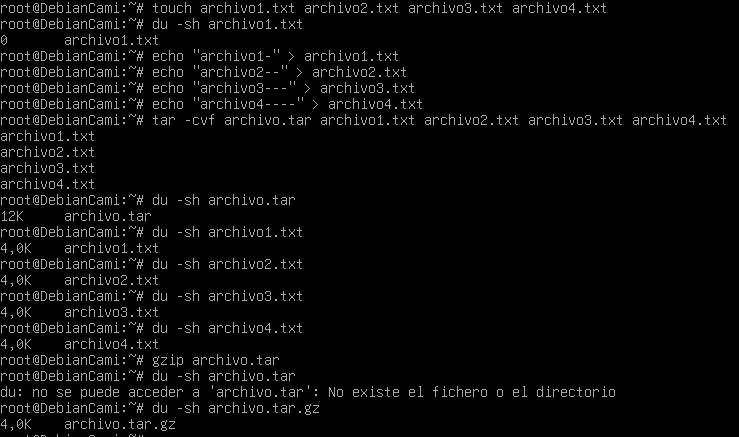
* killall -9 nombre\_proceso: Mata todos los procesos con el nombre especificado (SIGKILL).
* killall -u usuario: Termina todos los procesos de un usuario en particular.
* killall -I nombre\_proceso: Hace una búsqueda insensible a mayúsculas.

**top**: Muestra en tiempo real los procesos en ejecución, junto con información dinámica como uso de CPU, memoria, tiempo de ejecución, etc. Los datos se actualizan continuamente.

* top -u usuario: Filtra los procesos del usuario especificado.
* Dentro de top, se pueden usar teclas interactivas como k para matar procesos, r para reiniciarlos, y h para ayuda.
* top -n N: Muestra la salida de top N veces y luego termina.

**nice**: Ajusta la prioridad con la que un proceso se ejecuta. Un proceso con un valor de nice más bajo tiene más prioridad y acceso a recursos del sistema, mientras que uno con un valor más alto tiene menos prioridad.

* nice -n valor comando: Ejecuta un comando con la prioridad especificada (valor va de -20 a 19, donde -20 es la mayor prioridad).
* renice valor -p PID: Cambia la prioridad de un proceso en ejecución.
* 7)
  + El **empaquetado de archivos** en GNU/Linux hace referencia al proceso de agrupar varios archivos y/o directorios en un solo archivo comprimido o no comprimido, facilitando así su distribución, almacenamiento o transferencia. Este concepto es útil cuando se desea compartir o mover un conjunto de archivos de forma eficiente, o hacer copias de seguridad. Esto lo logramos usando tar.
  + Al empaquetar los 4 archivos en uno solo, lo único que hacemos es agruparlos (por ende, ocupan la suma de sus tamaños), mientras que si lo comprimimos, ocupan mucho menos.



* + Explicado en el ejercicio anterior.
  + Opción 1: tar -czvf archivos.tar.gz archivo1.txt archivo2.txt archivo3.txt archivo4.txt

Opción 2: tar -cjvf archivos.tar.bz2 archivo1.txt archivo2.txt archivo3.txt archivo4.txt

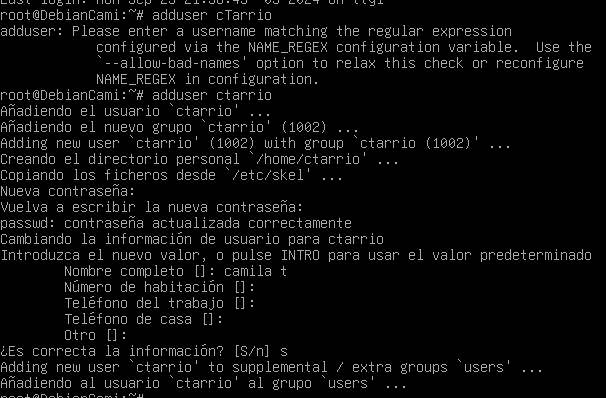
* + tar se utiliza para empaquetar (agrupar) varios archivos y directorios en un solo archivo, conocido como **archivo tar**, que mantiene la estructura de los archivos originales.

grep busca patrones de texto dentro de archivos o cadenas y muestra las líneas que coinciden con el patrón. Es muy útil para filtrar y buscar información en grandes volúmenes de texto.

gzip se usa para comprimir archivos. Es una herramienta de compresión que utiliza el algoritmo DEFLATE para reducir el tamaño de archivos individuales, generando archivos con la extensión .gz.

zgrep es similar a grep, pero permite buscar texto dentro de archivos que están comprimidos con gzip (.gz). Es una combinación de grep y gzip que facilita la búsqueda en archivos comprimidos sin tener que descomprimirlos manualmente.

wc cuenta el número de líneas, palabras y caracteres en un archivo o cadena. Es útil para obtener estadísticas sobre el contenido de un archivo o entrada.

* 8)
  + Análisis:
* ls –l > prueba: Genera un listado de todos los directorios y archivos del directorio de inicio del usuario de forma detallada y lo guarda en un archivo llamado prueba.
* ps > PRUEBA: Guarda un listado con la información de los procesos que se están ejecutando actualmente en un nuevo archivo llamado PRUEBA.
* chmod 710 prueba: Cambia los permisos de acceso al archivo prueba, le da permisos de lectura, escritura y ejecución al propietario, permiso de ejecución a los participantes de su grupo y cero permisos a los demás usuarios.
* chown root : root PRUEBA: Intenta cambiar el propietario y el grupo al que pertenece el archivo PRUEBA pero no puede porque no es un usuario root.
* chmod 777 PRUEBA: Cambia los permisos de acceso al archivo PRUEBA, le da permisos de lectura, escritura y ejecución al propietario, a los participantes de su grupo y a los demás usuarios.
* chmod 700 /etc/passwd: El usuario intenta cambiar los permisos de lectura, escritura y ejecución al directorio /etc/passwd pero esta operación no es posible porque sólo un usuario root puede administrar los permisos de los directorios almacenados en /etc.
* passwd root: El usuario intenta ver o cambiar la contraseña del usuario root pero esta acción no está permitida porque solo un usuario root puede cambiar la contraseña de otros usuarios.
* rm PRUEBA: Se elimina el archivo PRUEBA.
* man /etc/shadow: Esta acción no se puede realizar ya que el comando man no puede brindar un acceso a las páginas del manual del sistema ya que estas brindan información detallada sobre los comandos, utilidades, funciones del sistema y archivos del sistema operativo, no sobre directorios, además el permiso es denegado ya que intenta acceder al directorio shadow el cual contiene las contraseñas encriptadas de los usuarios.
* find / -name \*.conf: Busca y lista a partir del directorio raíz todos los archivos y directorios que contengan la extensión .conf.
* usermod root –d /home/newroot –L: El Usuario intenta cambiar el directorio de inicio del Usuario root a el directorio /home/newroot y además intenta bloquear su contraseña, esta acción no está permitida ya que el usuario no posee los permisos necesarios para ejecutar el comando.
* cd /root: El usuario intenta acceder al directorio de inicio del usuario root, algo no posible ya que el usuario no posee los permisos necesarios.
* rm \*: Se eliminan todos los archivos en el directorio donde está posicionado el usuario actualmente.
* cd /etc: El usuario se sitúa en el directorio de configuraciones del sistema.
* cp \* /home –R: El usuario intenta copiar todos los archivos y directorios del directorio actual /etc al directorio /home, incluyendo subdirectorios de manera recursiva pero esto no es posible ya que solo los usuarios root tienen permisos para copiar el contenido del directorio /etc.
* shutdown: El usuario apaga el sistema. Se necesitan permisos sudo para hacerlo.
* 9)
  + Se podrían utilizar los comandos “kill 23” ó “kill -9 23”.
  + El proceso llamado **init** es el proceso principal del sistema, con **PID 1**, que se encarga de la gestión y el control de otros procesos en el sistema. Utilizando el comando kill no se obtendría ningún resultado ya que el proceso init es de suma importancia para el arranque del sistema y este no puede ser terminado.
  + Se podría utilizar el comando “find / -type f –name \*.conf” este realizará una búsqueda desde el directorio raíz de todos los archivos que contengan la cadena “.conf” en su nombre, solo buscará archivos ya que este fue el tipo especificado para la búsqueda.
  + Se podría utilizar el comando “ps >/home/usuario/procesos”.
  + Se podría utilizar el comando “chmod 751 /home/usuario/xxxx”.
  + Se podría utilizar el comando “chmod 650 /home/usuario/yyyy”.
  + Se deberían de usar los comandos “cd /tmp” y luego “rm \*”.
  + Se podría utilizar el comando “sudo chown iso2021 /opt/isodata”.
  + Se podría utilizar el comando “pwd >> /home/usuario/donde”.
* 10)
  + Se podría realizar usando el comando “su” ó “su root”.
  + 
  + archivos modificados:

→ /etc/passwd

→ /etc/shadow

→ /etc/group

→ /etc/gshadow

directorios creados:

→ /home/ctarrio

* + comandos:
* cd /tmp
* mkdir cursada2017
  + comandos:
* cp -r /var/log/\* /tmp/cursada2017/
  + comandos:
* chown -R camila:users /tmp/cursada2017
  + comandos:
* chmod -R 723 /tmp/cursada2017
  + Ctrl + Alt + F2
  + comandos:
* tty

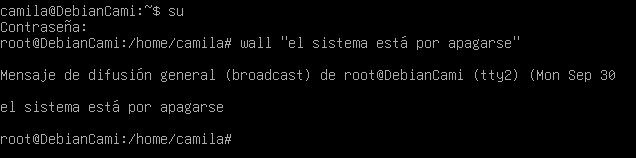
salida:

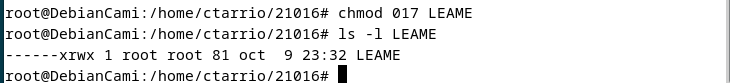
* /dev/tty2
  + comandos:
* ps –aux | wc -l

salida:

* 94
  + comandos:
* who -q

salida:

* root camila
* N° de usuarios = 2
  + 
  + comandos:
* shutdown -h now
* 11)
  + comandos:
* mkdir 21016
* cd 21016
  + comandos:
* vi LEAME
* i + \*ingresamos los datos\*
* esc + :wq
* cat LEAME (para verificar contenido)
  + comandos:
* chmod 017 LEAME



* + comandos:
* ls -la /etc
* ls -la /etc > ~/leame
* **LEAME** y **leame** son archivos distintos en Linux debido a la distinción entre mayúsculas y minúsculas.
  + El comando find es una herramienta poderosa que permite buscar archivos y directorios en el sistema de archivos en función de diferentes criterios como el nombre, tipo de archivo, fecha de modificación, permisos, etc.
* find [ruta] [opciones] [criterios]

Ej:

* find / -name "archivo.txt" (busca desde la raíz (/) por el nombre exacto (-name) el archivo con el nombre “archivo.txt”)
* find ~/ -name "\*.txt" (busca en el directorio de inicio del usuario actual (~/) todos los archivos con el patrón “\*.txt”)
  + comandos:
* find / -type f -name "\*.so" > ~/21016/.ejercicio\_f
* cat ~/21016/.ejercicio\_f
* 12)
  + Acciones:
* “mkdir iso”: Crea el directorio “iso” en su directorio inicial.
* “cd ./iso ; ps > f0”: Se sitúa en el directorio “iso” y crea el archivo “f0” con un listado de todos los procesos en ejecución en el sistema actualmente.
* “ls > f1”: Crea un archivo llamado “f1” en el directorio “iso” que contiene la lista de archivos y/o subdirectorios del directorio “iso”.
* “cd /”: Se sitúa en el directorio raíz.
* “echo $HOME”: Muestra en la salida estándar la dirección del directorio inicial del usuario.
* “ls –l &> $HOME/iso/ls”: Se crea un nuevo archivo llamado “ls” en el directorio “iso” ubicado en el directorio inicial del usuario que va a tener un listado detallado de los archivos y/o directorios del directorio raíz y además también la salida de error del comando realizado “ls –l”, esto se logra sumando a la redirección el símbolo “&”.
* “cd $HOME ; mkdir f2”: Se sitúa en su directorio inicial y crea el directorio “f2”.
* “ls –ld f2”: Muestra información detallada del directorio “f2”.
* Se especifica que se trate al mismo como un directorio con el parámetro “d”.
* “chmod 341 f2”: Cambia los permisos de lectura, escritura y ejecución del directorio “f2”. Escritura y Ejecución para el dueño, Lectura para el grupo y Ejecución para los demás usuarios.
* “touch dir”: Crea un archivo llamado “dir” en el directorio inicial del usuario.
* “cd f2”: Se sitúa en el directorio “f2”.
* “cd ~/iso”: Se sitúa en el directorio “iso”.
* “pwd > f3”: Crea un archivo nuevo en el directorio “iso” que contiene la información de la ruta actual del usuario en ese momento “~/iso”.
* “ps | grep “ ps ” | wc –l >> ../f2/f3”: Se listan todos los procesos en ejecución y el “grep” recibe esa lista como entrada estándar, allí filtra los resultados por aquellos procesos que posean la cadena “ps”, ese nuevo listado filtrado es pasado como entrada estándar al comando “wc” el cuál cuenta la cantidad de líneas de ese listado, es decir, la cantidad de procesos que se encontraron con la cadena “ps” para luego redirigir esa salida al archivo “f3” añadiendo la cantidad al final del mismo.
* “chmod 700 ../f2 ; cd ..”: Se modifican los permisos de lectura, escritura y ejecución del directorio “f2”. Lectura, Escritura y Ejecución para el dueño y cero permisos para el grupo y otros usuarios. Luego al ejecutar “cd ..” escala un nivel hacia arriba en la estructura de directorios, ahora situándose nuevamente en su directorio inicial.
* “find . –name etc/passwd”: Se muestra un mensaje de warning por el mal uso del comando “find”.
* “find / -name etc/passwd”: Muestra en pantalla el listado de los archivos y/o directorios de “/etc/passwd” ordenados por nombre.
* “mkdir ejercicio5”: Crea el directorio “ejercicio5”.
  + 
* 13)
* mkdir -p dir1/dir2 dir1/dir3/dir4 dir5
* touch dir1/dir2/f1 dir1/dir2/f2 dir1/dir3/f3 dir5/f4
  + comandos:
* mv ~/dir1/dir3/f3 ~/
* mkdir ~/dir11 -----------> cp ~/dir5/f4 ~/dir11/
* cp ~/dir5/f4 ~/dir11/f7
* mkdir ~/copia -----------> cp -r ~/dir1/\* ~/copia/
* mv ~/dir1/dir2/f1 ~/dir1/dir2/archivo -----------> ls -l ~/dir1/dir2/archivo
* chmod 763 ~/dir1/dir2/archivo
* mv ~/f3 ~/f3.exe -----------> mv ~/dir5/f4 ~/dir5/f4.exe
* chmod 023 ~/f3.exe ~/dir5/f4.exe
* 14)
  + comandos:
* sudo mkdir /tmp/logs
* sudo cp –a /var/log/\* /tmp/logs
* tar -cvf misLogs.tar -C /tmp logs
* tar -czvf misLogs.tar.gz -C /tmp logs
* cp misLogs.tar misLogs.tar.gz ~/
* rm -rf /tmp/logs
* tar xvf misLogs.tar -C 1 -----------> tar xvfz misLogs.tar.gz -C 2