

1)a) Podemos utilizar el vi, vim y mcedit.

b) vi, vim y mcedit son editores de texto en modo terminal que te permiten editar archivos de texto directamente desde la línea de comandos. Por otro lado, los comandos cat, more y less se utilizan para visualizar el contenido de archivos de texto.

Un editor de texto y los comandos cat, more y less tienen diferentes funciones y propósitos en sistemas Unix y Linux:

#### **Editor de Texto (como "vi" o "nano"):**

- Un editor de texto permite crear, editar y guardar archivos de texto.
- Proporciona un entorno interactivo para manipular y modificar el contenido de archivos.
- Permite realizar cambios en el texto, como agregar, eliminar y modificar contenido.
- El usuario puede trabajar en modo de inserción para agregar texto y en modo de comandos para realizar operaciones.
- Algunos ejemplos de editores de texto son "vi", "vim", "nano", "emacs" y "gedit".

#### **Comando cat:**

- El comando cat se utiliza para mostrar el contenido de archivos de texto en la terminal.
- Simplemente muestra el contenido completo del archivo en la salida estándar.
- Útil para ver rápidamente el contenido pequeño de archivos.

#### **Comando more:**

- El comando more permite visualizar el contenido de archivos de texto en la terminal, página por página.
- Permite desplazarse hacia adelante y hacia atrás en el contenido del archivo.
- Se utiliza presionando la tecla Espacio para avanzar una página y la tecla q para salir.
- Útil para leer contenido más extenso sin saturar la pantalla.

#### **Comando less:**

- El comando less es similar a more pero ofrece más características.
- Permite desplazarse hacia adelante y hacia atrás, además de buscar en el archivo.
- Utiliza las teclas de flecha, Espacio, Borrar y otras para la navegación.
- Es más flexible y útil para leer archivos más largos o realizar búsquedas en el contenido.

Los modos de operación del "vi" son 3:

**Modo de Inserción:** En este modo, puedes escribir y editar el contenido del archivo. Para entrar en este modo, presiona i o Insert.

**Modo Visual:** Este modo te permite seleccionar y manipular bloques de texto. Para entrar en este modo, presiona v.

**Modo de Comandos o Modo Normal:** En este modo, puedes realizar operaciones como moverte por el archivo, buscar, reemplazar, copiar y pegar, entre otras. Para entrar en este modo, presiona Esc.

c) Comandos útiles:

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

## Otros del chat gpt:

### Iniciar el Modo de Edición:

- Presiona la tecla i para entrar en el modo de inserción. Ahora puedes comenzar a escribir y editar el contenido del archivo.

### Escribir y Editar:

- En el modo de inserción, puedes escribir y editar el contenido del archivo como lo harías en cualquier otro editor de texto.

### Guardar Cambios:

- Para guardar los cambios que has realizado en el archivo, sigue estos pasos:  
Presiona la tecla Esc para salir del modo de inserción y asegurarte de estar en el modo de comandos.  
Escribe :w y presiona Enter. Esto guarda el archivo sin salir del editor.

### Salir del Editor:

- Para salir del editor "vi" después de guardar los cambios, sigue estos pasos:  
Asegúrate de estar en el modo de comandos presionando la tecla Esc.  
Escribe :q y presiona Enter. Esto cierra el editor si no hay cambios no guardados.

### Guardar y Salir en un Solo Paso:

- Para guardar los cambios y salir del editor en un solo paso, sigue estos pasos:  
Asegúrate de estar en el modo de comandos presionando la tecla Esc.  
Escribe :wq y presiona Enter. Esto guarda el archivo y cierra el editor.

### Salir sin Guardar Cambios:

- Si deseas salir del editor sin guardar los cambios realizados, sigue estos pasos:  
Asegúrate de estar en el modo de comandos presionando la tecla Esc.  
Escribe :q! y presiona Enter. Esto sale del editor sin guardar.

### Eliminar una Línea:

- Asegúrate de estar en el modo de comandos presionando la tecla Esc.
- Coloca el cursor sobre la línea que deseas eliminar.
- Escribe dd y presiona Enter. Esto eliminará la línea en la que se encuentra el cursor.

### Eliminar un Carácter:

- Asegúrate de estar en el modo de comandos presionando la tecla Esc.
- Coloca el cursor sobre el carácter que deseas eliminar.
- Presiona x. Esto eliminará el carácter bajo el cursor.

#### **Insertar Nueva Línea:**

- Asegúrate de estar en el modo de comandos presionando la tecla Esc.
- Coloca el cursor en la posición donde deseas insertar la nueva línea.
- Escribe o para abrir una nueva línea debajo del cursor o O para abrir una nueva línea arriba del cursor.

2)a)

1 Se empieza a ejecutar el código del BIOS

2 El BIOS ejecuta el POST

3 El BIOS lee el sector de arranque (MBR)

4 Se carga el gestor de arranque (MBC)

5 El bootloader carga el kernel y el initrd

6 Se monta el initrd como sistema de archivos raíz y se inicializan componentes esenciales (ej.: scheduler)

7 El Kernel ejecuta el proceso init y se desmonta el initrd

8 Se lee el /etc/inittab

9 Se ejecutan los scripts apuntados por el runlevel 1

10 El final del runlevel 1 le indica que vaya al runlevel por defecto

11 Se ejecutan los scripts apuntados por el runlevel por defecto

12 El sistema está listo para usarse

b) Lo ejecuta el kernel. "init" es el primer proceso que se inicia en sistemas Unix y sistemas basados en Unix, como Linux. Su objetivo principal es inicializar el sistema operativo y coordinar la ejecución de otros procesos esenciales durante el proceso de arranque. "init" es el padre de todos los procesos en el sistema y se inicia con un ID de proceso (PID) de 1 (chattgpt).

c) Si ejecutas el comando pstree en la terminal de Linux con Debian, obtendrás una representación jerárquica de los procesos en ejecución en forma de un árbol. Esta representación mostrará la relación entre los procesos y sus procesos hijos, lo que te permitirá comprender mejor cómo están organizados y cómo interactúan en el sistema. La salida de pstree mostrará todos los procesos en ejecución junto con su relación jerárquica. Cada proceso se representa por su nombre y su ID de proceso (PID), y los procesos hijos se indentan debajo de sus padres, lo que facilita la visualización de la estructura de procesos del sistema.

d) Runlevels:

Los niveles de ejecución (RunLevels en inglés) son estados predefinidos en los que puede encontrarse un sistema operativo Unix o Linux. Cada nivel de ejecución tiene un conjunto específico de servicios y procesos que se inician o detienen, lo que define el estado y la funcionalidad del sistema en ese momento. Los niveles de ejecución son una parte fundamental del proceso de arranque y administración de un sistema, **ya que permiten que**

**el sistema funcione en diferentes modos, desde el arranque hasta el apagado y la recuperación.**

En sistemas Unix y Linux, los niveles de ejecución generalmente se identifican con números o letras. Los números son más comunes y suelen variar desde 0 hasta 6, aunque la cantidad de niveles y su significado específico pueden variar según la distribución del sistema.

**Aquí está el significado general de algunos niveles de ejecución comunes:**

- Nivel 0 (Halt/Shutdown): Se utiliza para apagar el sistema. En este nivel, se detienen todos los servicios y el sistema se apaga de manera segura.
- Nivel 1 (Single User/Recovery): También conocido como modo de usuario único o recuperación. Se utiliza para el mantenimiento y la recuperación del sistema, ya que solo se inician servicios esenciales.
- Niveles 2-5: En muchas distribuciones, estos niveles son idénticos y no tienen un significado específico predefinido. Los administradores del sistema pueden personalizar estos niveles según sus necesidades.
- Nivel 6 (Reboot): Se utiliza para reiniciar el sistema. Al igual que en el nivel 0, se detienen todos los servicios antes de reiniciar.

**El objetivo de los niveles de ejecución es proporcionar modos específicos de funcionamiento del sistema que sean adecuados para diferentes situaciones.** Por ejemplo:

- Arranque Normal: El sistema arranca en un nivel que inicia todos los servicios necesarios para un funcionamiento normal, como el entorno gráfico, la red y otros servicios.
- Mantenimiento o Recuperación: El sistema puede arrancar en un nivel de recuperación en caso de problemas graves que requieran correcciones manuales.
- Apagado: El sistema puede apagar servicios y sistemas antes de apagarse completamente.

e) lo primero está arriba.

La definición de qué nivel de ejecución (runlevel) se ejecutará al iniciar el sistema operativo se encuentra **en el archivo de configuración del sistema de inicialización**. La ubicación y el nombre de este archivo pueden variar según la distribución de Linux que estés utilizando. Sin embargo, en muchas distribuciones modernas, este archivo suele ser `/etc/inittab`.

El archivo `/etc/inittab` (o su equivalente en sistemas con `systemd`) contiene configuraciones relacionadas con la inicialización del sistema, incluida la configuración del nivel de ejecución predeterminado. En este archivo, encontrarás una línea similar a esta:

```
id:3:initdefault:
```

En esta línea, el número 3 representa el nivel de ejecución predeterminado. En este ejemplo, el nivel de ejecución predeterminado es el nivel 3.

No todas respetan estos estándares.

f)

El archivo **/etc/inittab** tiene un papel central en los sistemas Unix y Linux que utilizan el sistema de inicialización SysVinit. **Este archivo contiene información importante sobre cómo se deben iniciar y gestionar los procesos y servicios durante el proceso de arranque del sistema.** Aunque en sistemas más modernos con systemd este archivo puede tener menos relevancia, en sistemas que aún utilizan SysVinit, sigue siendo esencial. La finalidad del archivo **/etc/inittab** es proporcionar instrucciones al sistema sobre cómo se deben configurar los niveles de ejecución (runlevels) y cómo se deben iniciar los diferentes procesos y servicios en cada nivel. El archivo contiene líneas que definen las acciones que se deben realizar durante el proceso de arranque y en respuesta a ciertos eventos. La estructura de las líneas en el archivo **/etc/inittab** es generalmente la siguiente:

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

- id: Identificador único para la entrada.
- runlevels: Lista de niveles de ejecución a los que se aplica la entrada.
- action: Acción que se realizará para la entrada. Puede ser "initdefault" (nivel de ejecución predeterminado), "respawn" (iniciar de nuevo si se detiene), "wait" (esperar), etc.
- process: El proceso o comando que se ejecutará según la acción especificada.

Un ejemplo de una línea en **/etc/inittab** podría ser:

```
id:3:initdefault:
```

En esta línea, se establece que el nivel de ejecución predeterminado es el nivel 3. Es importante mencionar que en distribuciones más modernas que utilizan systemd, la relevancia del archivo **/etc/inittab** ha disminuido, ya que systemd adopta un enfoque diferente para la gestión de servicios y niveles de ejecución.

g)

Para cambiar del runlevel <X> al runlevel <Y> en sistemas que utilizan el sistema de inicialización SysVinit (no systemd), puedes utilizar el comando **init**. Sin embargo, ten en cuenta que cambiar el runlevel mediante **init** solo es efectivo para la sesión actual y no es permanente.

Los pasos para cambiar de runlevel son los siguientes:

Abre una terminal.

Ejecuta el siguiente comando para cambiar al runlevel <Y>:

```
bash
```

Copy code

```
sudo init <Y>
```

Donde <Y> es el número del runlevel al que deseas cambiar.

El sistema realizará la transición al runlevel <Y>, deteniendo los servicios y procesos correspondientes al runlevel <X> y activando los servicios y procesos relacionados con el runlevel <Y>.

Es importante destacar que este cambio no es permanente. La próxima vez que reinicies el sistema, volverá al runlevel predeterminado configurado **en el archivo /etc/inittab o en la configuración del sistema**. Si deseas cambiar el runlevel predeterminado de manera permanente, deberás modificar la configuración en el archivo **/etc/inittab** o utilizar

herramientas específicas para administrar niveles de ejecución, dependiendo de la distribución que estés utilizando.

#### h) **CONSULTAR**

Los scripts RC (Scripts de Control de Arranque) son scripts utilizados en sistemas Unix y Linux para gestionar el proceso de arranque (inicio) y detención (apagado) del sistema operativo. Estos scripts tienen la finalidad de iniciar o detener servicios, configurar el sistema y realizar otras tareas necesarias durante el proceso de arranque o apagado del sistema.

En sistemas que utilizan el sistema de inicialización SysVinit, como muchas distribuciones Linux tradicionales, los scripts RC se almacenan en directorios específicos, generalmente en `/etc/rc.d/` o `/etc/init.d/`. Cada script RC corresponde a un servicio o proceso en particular y contiene instrucciones para iniciar o detener ese servicio según corresponda.

Cuando un sistema GNU/Linux arranca o se detiene, se ejecutan los scripts RC pertinentes según el nivel de ejecución (runlevel) en el que se encuentra el sistema. El sistema determina qué script ejecutar basándose en la información almacenada en el archivo `/etc/inittab` o en el sistema de control de servicios en uso (por ejemplo, `systemd`).

Los niveles de ejecución (runlevels) indican en qué modo se encuentra el sistema (inicio, apagado, modo de usuario único, etc.). Cada nivel de ejecución está asociado con un conjunto específico de servicios que deben iniciarse o detenerse. El sistema consulta el archivo `/etc/inittab` (o la configuración equivalente en sistemas modernos) para determinar qué servicios deben iniciarse o detenerse en cada nivel de ejecución. Luego, se ejecutan los scripts RC correspondientes para realizar estas acciones.

En cuanto al orden para llamar a los scripts RC, generalmente se sigue una convención en la que los scripts se numeran con un prefijo para indicar el orden en el que deben ejecutarse. Por ejemplo, los scripts que inician servicios esenciales pueden tener números bajos (por ejemplo, `S10network`), mientras que los scripts que detienen servicios pueden tener números altos (por ejemplo, `K90apache`). Esto garantiza que los servicios se inicien en el orden correcto y se detengan en el orden inverso al apagar el sistema.

#### i)

**insserv** es una herramienta utilizada en sistemas Linux que utilizan el sistema de inicialización SysVinit para administrar los scripts de inicio y detención de servicios. Su función principal es determinar el orden en que los scripts de inicialización deben ejecutarse al arrancar o apagar el sistema, siguiendo las dependencias entre los servicios.

La utilización de **insserv** es una mejora respecto al arranque tradicional en el sentido de que automatiza y simplifica el proceso de definir el orden de arranque de los servicios. Aquí hay algunas ventajas que **insserv** proporciona en comparación con un arranque tradicional:

**Gestión de Dependencias:** **insserv** analiza las dependencias entre los scripts de inicio y detención de servicios para asegurarse de que los servicios se inicien en el orden correcto y se detengan en el orden inverso. Esto es esencial para garantizar que los servicios que dependen de otros estén disponibles cuando se inician y se detengan en el orden adecuado durante el apagado.

**Simplificación de la Configuración:** **insserv** utiliza encabezados en los scripts de inicialización para definir las dependencias y el orden de ejecución. Esto simplifica la configuración en comparación con tener que asignar manualmente números de secuencia a los scripts.

Fácil Mantenimiento: Al utilizar encabezados en los scripts de inicialización, es más fácil agregar, modificar o eliminar dependencias entre los servicios. Esto simplifica el mantenimiento del sistema y evita posibles conflictos entre los scripts.

Mayor Flexibilidad: insserv permite manejar de manera más efectiva situaciones complejas de dependencias entre servicios, lo que puede ser difícil de lograr con un enfoque tradicional.

En resumen, insserv es una herramienta que automatiza la gestión de dependencias y el orden de ejecución de los scripts de inicio y detención de servicios en sistemas SysVinit. Proporciona una forma más estructurada y eficiente de configurar el arranque y apagado del sistema, mejorando la integridad y la administración de los servicios.

j) (lo dijo el profe en un video de explicacion 2) Una desventaja del proceso SystemV es que bloqueaba futuras tareas hasta que se termine la actual (es sincrónico). Por esto y otras desventajas, se fueron buscando alternativas nuevas. Esta fue **Upstart**, este permite la ejecución de trabajos en forma asincrónica a través de eventos. A grande rasgos, en vez de que los scripts se ejecuten en un runlevel específico, ahora se van a ejecutar en el momento en que un evento suceda. Cada job va a estar definido en el /etc/init y pueden ser de 2 tipos: task (tarea) o service (servicio).

Upstart es compatible con System V.

Cada job tiene un objetivo que es de inicio o de parada y un estado en el cual se encuentra. Puede tener uno o varias tareas ejecutables como parte de su ciclo de vida y siempre debe existir la tarea principal.

Acá no hay mas /etc/inittab.

k) **CONSULTAR MÁS** SystemV es sincrónico y Upstart asincrónico. No hay más /etc/inittab en Upstart.

CHAT GPT:

Historia y Origen:

- System V: Es uno de los sistemas de inicialización más antiguos y ampliamente utilizados. Se basa en una serie de scripts de inicialización almacenados en directorios específicos. Es el sistema de inicialización predeterminado en muchas distribuciones Linux tradicionales.
- Upstart: Fue desarrollado por Canonical, la empresa detrás de Ubuntu, como un intento de mejorar y modernizar el sistema de inicialización. Introdujo cambios significativos en la forma en que se gestionan los servicios y procesos.

**Configuración y Scripts:**

- System V: Utiliza scripts de inicialización **(los RC?)** almacenados en directorios como /etc/init.d/ para controlar los servicios y procesos. Los scripts suelen tener nombres que indican su función y se pueden configurar con niveles de ejecución específicos.
- Upstart: Utiliza archivos de configuración en formato .conf **(SE REFIERE A LOS "JOBS"?)** ubicados en /etc/init/. Estos archivos son más declarativos y permiten definir cómo se inician y administran los servicios, incluidas las dependencias.

Gestión de Dependencias:



- System V: La gestión de dependencias entre servicios puede ser más compleja y requiere asignar números de secuencia para controlar el orden de inicio y detención.
- Upstart: Proporciona una gestión de dependencias más avanzada. Los archivos de configuración permiten especificar claramente las dependencias de cada servicio, lo que facilita la definición y el control del orden de inicio.

#### Compatibilidad:

- System V: Debido a su larga historia, muchos sistemas y aplicaciones están diseñados para trabajar con scripts de inicialización de System V. Por lo tanto, es más compatible con software heredado.
- Upstart: Aunque Ubuntu adoptó Upstart, otros sistemas no lo adoptaron de manera tan amplia, y finalmente fue reemplazado por systemd en muchas distribuciones.

#### Flexibilidad:

- System V: Puede ser menos flexible para manejar situaciones complejas de arranque y apagado.
- Upstart: Ofrece mayor flexibilidad para definir comportamientos y acciones personalizadas durante el arranque y apagado del sistema.

#### Adopción:

- System V: Fue el sistema de inicialización predominante durante mucho tiempo, pero ha sido reemplazado gradualmente por systemd en muchas distribuciones modernas.
- Upstart: Aunque tuvo cierta adopción, especialmente en Ubuntu, su uso no fue tan generalizado y fue reemplazado por systemd en la mayoría de las distribuciones.

L) Lo reemplaza los llamados "jobs" o archivos de configuración. Se encuentran en /etc/init/.

m)

- # MySQL Service: Un comentario que describe el servicio que se está configurando.
- description "MySQL Server": Define una descripción del servicio que se mostrará en los registros y en las herramientas de administración. En este caso, el servicio se llama "MySQL Server".
- author "infoautor": Un comentario que podría contener información sobre el autor o cualquier otra información relevante.
- start on (net-device-up and local-filesystems and runlevel [2345]): Esta línea especifica cuándo debe iniciarse el servicio. En este caso, el servicio debe iniciarse cuando la red esté activa (net-device-up), los sistemas de archivos locales estén montados (local-filesystems) y el sistema esté en uno de los niveles de ejecución 2, 3, 4 o 5 (runlevel [2345]).
- stop on runlevel [016]: Indica en qué niveles de ejecución debe detenerse el servicio. En este caso, el servicio se detendrá en los niveles de ejecución 0, 1 y 6.
- exec /usr/sbin/mysqld: Esta línea especifica el comando que se debe ejecutar para iniciar el servicio. En este caso, se ejecutará el comando /usr/sbin/mysqld, que es el comando para iniciar el servidor de MySQL.



Las secciones [...] representan partes omitidas del archivo de configuración que podrían contener más configuraciones específicas para el servicio de MySQL. En general, este archivo de configuración define cómo se inicia, detiene y administra el servicio de base de datos MySQL utilizando el sistema de inicialización Upstart.

n) (sin chat gpt)

Otra alternativa al SystemV y Upstart, es asincrónico al igual que el upstart y en vez de trabajar en base a runlevels trabaja en base a **targets**, estos últimos pueden ser activados o desactivados a través del comando **systemctl**. Al igual que Upstart, puede mantener compatibilidad con System V. Acá tampoco existe el /etc/inittab.

Tanto Systemd como Upstart como SystemV son sistemas de inicialización (gpt), uno más moderno que el otro.

Systemd va a estar manejando un conjunto de **units** y targets, estos units realizarán ciertas tareas, a diferencia de el system V la cual tenias scripts de start and kill, acá hay units de diversos tipos:

**Service:** controla un servicio particular (.service)

**Socket:** Permiten comunicarse entre units o activar y desactivar units a partir de ciertos eventos.

**Target:** agrupa units o establece puntos de sincronización durante el booteo (.target)  
→ dependencia de unidades

**Snapshot:** almacena el estado de un conjunto de unidades que puede ser restablecido más tarde (.snapshot).

Los **unit** pueden tener 2 estados, **active o inactive**.

En systemd podemos desactivar o activar servicios bajo demanda por medio de **sockets**.

ñ) (sin chat gpt) Hace referencia de que en systemd podemos desactivar o activar servicios bajo demanda por medio de **sockets**. Cada servicio tiene conectado un socket que este recibe una señal y activa a dicho servicio.

o) (sin chat gpt) **cgroups:** Permite organizar un grupo de procesos en forma jerárquica. Agrupa un conjunto de procesos relacionados. Tareas que realiza:

- Tracking mediante subsistema cgroups (no se usa el PID).
- Limita el uso de recursos.

3) a) Estos son: (sin chat gpt).

**/etc/passwd** vas a tener a todos los usuarios con sus atributos.

El archivo está estructurado de tal manera en el cual cada línea es un usuario, en cada línea vas a encontrar todos los atributos del usuario. (ej id del usuario, nombre de usuario, etc).

**/etc/group** te muestra los grupos secundarios del sistema.

**/etc/shadow** te dice el nombre del usuario y la contraseña. solo puede acceder por el usuario root o superusuario y está encriptada.

b)

Las siglas UID y GID se refieren a los siguientes conceptos:

- UID (User Identifier): El UID es un número único que identifica de manera exclusiva a un usuario en un sistema GNU/Linux. Cada usuario tiene su propio UID, y es utilizado internamente por el sistema operativo para identificar y gestionar las diferentes cuentas de usuario.
- GID (Group Identifier): El GID es un número único que identifica de manera exclusiva a un grupo en un sistema GNU/Linux. Los grupos permiten organizar a los usuarios en categorías lógicas para facilitar la administración de permisos y accesos.

En cuanto a si pueden coexistir UIDs iguales en un sistema GNU/Linux, la respuesta es que no deberían coexistir. Los UIDs son utilizados para distinguir y gestionar las cuentas de usuario, y un UID único es esencial para garantizar que cada cuenta se maneje de manera individual.

Si dos usuarios tuvieran el mismo UID, se crearían conflictos y ambigüedades en el sistema operativo. Los permisos y accesos podrían asignarse incorrectamente, lo que llevaría a problemas de seguridad y administración. Cada vez que se intenta acceder a un archivo o recurso, el sistema utiliza el UID para verificar los permisos y determinar si el usuario tiene los derechos adecuados.

c)

El usuario "root" es el superusuario en los sistemas operativos Unix y Linux. También se conoce como el "usuario administrador". El usuario root tiene acceso completo y sin restricciones a todos los recursos y comandos del sistema, lo que le permite realizar cambios críticos en la configuración, instalar y desinstalar software, administrar usuarios y grupos, y realizar otras tareas que requieren privilegios de administración.

En un sistema GNU/Linux, normalmente solo existe un único usuario "root". Aunque técnicamente sería posible crear más de un usuario con el nombre "root", esto no es recomendable y puede causar confusiones y problemas de seguridad. El usuario root es único y se utiliza para realizar tareas de administración que requieren los máximos privilegios.

La UID del usuario root suele ser 0. La UID (User Identifier) es un número que identifica de manera única a un usuario en el sistema. La UID 0 se reserva para el usuario root en la mayoría de los sistemas Unix y Linux. Esto significa que cuando se verifica la UID de un usuario, si es 0, se trata del usuario root.

d)

Primero pones "su" para iniciar sesión en un usuario que sea superusuario.

Después tienes que crear el grupo: `sudo groupadd catedra`.

después aplicas este comando: `"sudo adduser iso2017 --home /home/iso_2017 --ingroup catedra"` Lo que hace es:

`sudo`: El comando `sudo` se utiliza para ejecutar el comando con privilegios de superusuario (root). Requiere que el usuario que lo ejecute tenga permisos para realizar operaciones administrativas.

- adduser: Este es el comando que se utiliza para agregar un nuevo usuario al sistema.
- iso2017: Es el nombre del usuario que deseas crear.
- --home /home/iso\_2017: Este parámetro especifica la ubicación del directorio de inicio (home) del nuevo usuario. En este caso, estás configurando el directorio de inicio en /home/iso\_2017. El directorio se creará si no existe.
- --ingroup catedra: Con este parámetro, estás agregando al usuario al grupo llamado "catedra". Si el grupo no existe, el comando lo creará automáticamente. Esto significa que el usuario será miembro de este grupo.

En resumen, el comando completo realiza las siguientes acciones:

Utiliza sudo para ejecutar el comando con privilegios de superusuario.

Crea un nuevo usuario con el nombre "iso2017".

Establece el directorio de inicio del usuario en /home/iso\_2017.

Agrega al usuario al grupo "catedra". Si el grupo no existe, lo crea.

Le tienes que agregar una contraseña al grupo.

Configuras la información de usuario para iso 2017.

vas al /home y ves que tienes la carpeta iso\_2017 .

entras ahí con el cd /home/iso+tab.

creas un archivo (te pide que le crees un archivo en su home personal). "touch hola.txt".

Cargale algo con "vi hola.txt".

Ahora si te fijas con "ls -l hola.txt" ves los permisos, el primero hace referencia al dueño y el segundo al grupo de ese archivo, y si te fijas, como no iniciaste sesión desde el usuario creado (iso2017) sino desde el superusuario, en mi caso el "root" te figura en ambos que tienes el root, entonces le tienes que asignar el nuevo dueño y el nuevo grupo creado (iso2017 y catedra) para eso haces esto:

"sudo chgrp catedra (el grupo) hola.txt". te cambia el grupo.

y después "sudo chown iso2017 (usuario) hola.txt" y te cambia el dueño.

entonces ahora no cualquier usuario puede acceder a ese archivo, ya que es único para ese dueño y ese grupo catedra.

e)

useradd/ adduser:

- Funcionalidad: Estos comandos se utilizan para agregar un nuevo usuario al sistema.
- Diferencia: useradd es el comando básico, mientras que adduser es un script interactivo que hace uso de useradd pero ofrece una experiencia más amigable para agregar usuarios, como permitirte establecer contraseñas y agregar al usuario a grupos automáticamente.
- Ejemplo: sudo adduser nuevo\_usuario

usermod:

- Funcionalidad: Permite modificar las propiedades de un usuario existente.
- Ejemplo: sudo usermod -aG grupo usuario (para agregar un usuario a un grupo)

userdel:

- Funcionalidad: Elimina un usuario del sistema.
- Ejemplo: sudo userdel usuario

su (Switch User):

- Funcionalidad: Cambia al usuario superusuario (root) o a otro usuario. Sin argumentos, cambia a root. Con argumentos, cambia al usuario especificado.
- Ejemplo: su usuario

groupadd:

- Funcionalidad: Agrega un nuevo grupo al sistema.
- Ejemplo: sudo groupadd nuevo\_grupo

who:

- Funcionalidad: Muestra información sobre los usuarios que están actualmente conectados al sistema.
- Ejemplo: who

groupdel:

- Funcionalidad: Elimina un grupo del sistema.
- Ejemplo: sudo groupdel grupo

passwd:

- Funcionalidad: Cambia la contraseña de un usuario.
- Ejemplo: sudo passwd usuario (cambia la contraseña del usuario actual)

4)

- a) **Los permisos** se pueden aplicar a archivos y directorios (son permisos para el archivo), **existen de lectura, escritura y ejecución (están en notación octal)**. Estos permisos actúan sobre los usuarios, pueden estar aplicados al dueño del archivo, al grupo del archivo o cualquier otro usuario. Los primeros permisos están en valor octal.

Para asignarle permisos a un directorio o archivo se usa el comando **chmod**. Un archivo tiene un usuario (el dueño) y un grupo.

Ej: \$ chmod 755 /tmp/script. En este código se le aplica al dueño del usuario (al primer número) el valor 7 que hace incapié a los permisos de lectura, escritura y ejecución. Luego al segundo número que sería el grupo del archivo se le aplica los permisos de lectura y ejecución (al igual que en el tercer número).

**Si aplicas ls -l te dice la info sobre los archivos y directorios en el lugar donde te encuentras:**

**ej:**

```
-rw-r--r-- 1 usuario grupo 1234 ago 26 10:00 archivo.txt
drwxr-xr-x 2 usuario grupo 4096 ago 26 09:30 carpeta
```

El primer campo muestra los permisos del archivo o directorio. Los permisos se dividen en tres grupos: propietario, grupo y otros. Los caracteres -, r, w y x indican si el archivo es un archivo regular (-) o un directorio (d), y si el propietario, el grupo y otros tienen permisos de lectura (r), escritura (w) y ejecución (x).

- b) el comando chmod lo dije arriba. **(para el parcial usa esta, pero sino funciona y es mas comodo la de aca abajo).**

### 1. **``chmod``:**

El comando `chmod` se utiliza para cambiar los permisos de acceso a archivos y directorios en sistemas Unix-like, incluyendo GNU/Linux. Puedes usar `chmod` para asignar o quitar permisos de lectura, escritura y ejecución a propietarios, grupos y otros usuarios.

Algunos parámetros comunes de `chmod` son:

- `u` (propietario)
- `g` (grupo)
- `o` (otros)
- `a` (todos)
- `+` (agregar permisos)
- `-` (quitar permisos)
- `=` (establecer permisos específicos)

Por ejemplo, para dar permisos de lectura y escritura al propietario de un archivo:

```
...
chmod u+rw archivo.txt
...
```

### 2. **``chown``:**

El comando `chown` se utiliza para cambiar el propietario y/o grupo de un archivo o directorio. Esto es útil para transferir la propiedad de un archivo a otro usuario o grupo.

Algunos ejemplos de uso de `chown` son:

- Cambiar el propietario de un archivo:

```
...
chown nuevo_propietario archivo.txt
...
```

- Cambiar el propietario y el grupo de un archivo:

```
...
chown nuevo_propietario:nuevo_grupo archivo.txt
...
```

### 3. **``chgrp``:**

El comando `chgrp` se utiliza para cambiar el grupo propietario de un archivo o directorio. Esto te permite cambiar la asociación de grupo sin cambiar el propietario.

Ejemplo de uso:

```
...
chgrp nuevo_grupo archivo.txt
...
```

Estos comandos son esenciales para administrar y configurar los permisos de archivos y directorios en un sistema GNU/Linux. Pueden ser utilizados con precaución ya que los permisos incorrectos pueden afectar la seguridad y el funcionamiento del sistema. Puedes obtener más información detallada sobre estos comandos utilizando las páginas de manual, por ejemplo, ``man chmod``, ``man chown`` y ``man chgrp``.

- c) lo explique antes.
- d) No se debería poder acceder a archivos sin permisos, sin embargo si sos superusuario y tenes permisos de poder realizar cualquier tipo de cambios puedes cambiar los permisos a un archivo y te pones como dueño por ejemplo. Otra es que apliques el comando `sudo cat` para acceder al contenido de ese archivo, ya que si sos superusuario tambien puedes.

e) Los conceptos de "full path name" (nombre de ruta completa) y "relative path name" (nombre de ruta relativa) son términos utilizados en sistemas de archivos para especificar la ubicación de un archivo o directorio dentro de la estructura de directorios del sistema.

#### 1. **\*\*Full Path Name (Nombre de Ruta Completa):\*\***

Un "full path name" es una ruta que describe la ubicación completa de un archivo o directorio desde la raíz del sistema de archivos. Incluye todos los directorios padres, separados por barras diagonales ("/"), hasta llegar al archivo o directorio en cuestión.

Por ejemplo, si tienes un archivo llamado "documento.txt" dentro de la carpeta "proyectos" que está dentro de tu directorio personal, la ruta completa podría ser algo como:

```
...  
/home/tu_usuario/proyectos/documento.txt  
...
```

#### 2. **\*\*Relative Path Name (Nombre de Ruta Relativa):\*\***

Un "relative path name" es una ruta que describe la ubicación de un archivo o directorio en relación con el directorio actual en el que te encuentras. No parte desde la raíz del sistema de archivos, sino que parte del directorio en el que estás trabajando.

Por ejemplo, si estás actualmente en el directorio `/home/tu_usuario/`, y quieres acceder al archivo "documento.txt" en la carpeta "proyectos" dentro de tu directorio personal, la ruta relativa podría ser:

```
...  
proyectos/documento.txt  
...
```

O si estás dentro del directorio `/home/tu_usuario/proyectos/`, la ruta relativa podría ser simplemente:

```
...  
documento.txt  
...
```

En resumen, un "full path name" especifica la ubicación completa de un archivo o directorio desde la raíz del sistema de archivos, mientras que un "relative path name" se refiere a la ubicación relativa en relación con el directorio actual en el que te encuentras. Los nombres de ruta relativa son especialmente útiles cuando deseas referenciar archivos o directorios en función de tu ubicación actual sin tener que escribir toda la ruta desde la raíz del sistema.

f) Con el pwd.

Sí, en sistemas Unix-like, incluyendo GNU/Linux, existen formas de acceder a tu directorio personal sin necesidad de escribir la ruta completa. Esto se logra utilizando el carácter especial `~` (tilde), que se expande automáticamente al directorio personal del usuario actual. Esta idea también se puede aplicar para acceder a los directorios de otros usuarios si tienes permisos para acceder a ellos.

Aquí tienes un ejemplo de cómo funciona:

1. **\*\*Acceder a tu Directorio Personal:\*\***

Si estás en cualquier ubicación en el sistema y deseas acceder a tu directorio personal, puedes usar el tilde `~`. Por ejemplo, si estás en el directorio `"/home/tu_usuario/proyectos/"` y quieres ir a tu directorio personal:

```
...  
cd ~  
...
```

Esto te llevará a tu directorio personal, que es `"/home/tu_usuario/"`.

2. **\*\*Acceder al Directorio Personal de Otro Usuario:\*\***

Si tienes permisos para acceder a los directorios de otros usuarios, puedes utilizar el tilde `~` seguido del nombre de usuario para acceder a su directorio personal. Por ejemplo, si deseas acceder al directorio personal del usuario `"otro_usuario"`:

```
...  
cd ~otro_usuario  
...
```

Esto te llevará al directorio personal del usuario `"otro_usuario"`.

El uso del tilde (`~`) para acceder a directorios personales o de otros usuarios es una manera conveniente de evitar tener que escribir rutas completas largas. Sin embargo, ten en cuenta que solo podrás acceder a directorios de otros usuarios si tienes los permisos adecuados para hacerlo.

g)

Aquí tienes información sobre los comandos relacionados con el uso del sistema de archivos en GNU/Linux:



1. **`cd`:`** Cambia el directorio actual. Puedes usarlo sin argumentos para ir al directorio personal del usuario o proporcionar una ruta para moverte a otro directorio.
2. **`umount`:`** Desmonta un sistema de archivos montado previamente. Puedes usarlo para desconectar dispositivos, como unidades USB o particiones.
3. **`mkdir`:`** Crea un nuevo directorio en la ubicación especificada.
4. **`du`:`** Muestra el uso del espacio en disco de archivos y directorios. Puede usarse con diferentes opciones para obtener información detallada.
5. **`rmdir`:`** Elimina un directorio vacío.
6. **`df`:`** Muestra el espacio en disco disponible y utilizado en sistemas de archivos montados. Puedes usarlo para verificar la cantidad de espacio libre en tus dispositivos.
7. **`mount`:`** Monta un sistema de archivos en un directorio para que sea accesible en ese punto. Es comúnmente usado para montar dispositivos como unidades USB o particiones.
8. **`ln`:`** Crea enlaces simbólicos o enlaces duros a archivos. Un enlace simbólico es un atajo a otro archivo, mientras que un enlace duro crea múltiples nombres de archivo que apuntan al mismo contenido.
9. **`ls`:`** Lista los archivos y directorios en el directorio actual. Puede usarse con opciones para mostrar detalles como permisos, propietario, fecha y tamaño.
10. **`pwd`:`** Muestra la ruta completa del directorio actual.
11. **`cp`:`** Copia archivos o directorios de una ubicación a otra.
12. **`mv`:`** Mueve (cambia de ubicación) archivos o directorios. También se usa para cambiar nombres.

5)

a)

Un proceso en un sistema operativo es una entidad en ejecución que representa la ejecución de un programa o una tarea en particular. Cada proceso tiene su propio espacio de memoria, recursos asignados y estado de ejecución. Los procesos son la unidad básica de ejecución en un sistema operativo y permiten que múltiples tareas se ejecuten de manera concurrente y aislada.

- **PID (Process ID):** Las siglas PID se refieren al "Process ID" o "Identificador de Proceso". Es un número único asignado a cada proceso en el sistema operativo. Los PID son utilizados para identificar y gestionar los procesos. Cada vez que se crea un nuevo proceso, se le asigna un PID único.

- **PPID (Parent Process ID):** Las siglas PPID se refieren al "Parent Process ID" o "Identificador de Proceso Padre". Indica el PID del proceso que creó el proceso actual. En

otras palabras, es el PID del proceso padre que generó el proceso actual. Esto es útil para rastrear la jerarquía de procesos.

En GNU/Linux, todos los procesos tienen atributos como PID y PPID. Estos son atributos esenciales que se utilizan para la gestión de procesos y la organización de la jerarquía de procesos en el sistema.

**LOS ATRIBUTOS TMB LO PODES VER CON EL COMANDO “TOP” EN LA TERMINAL (ALGUNOS), ME DIJO EL PROFE QUE A LO QUE SE REFIERE ESOS ATRIBUTOS SERIA LO DEL PSB, PERO NO ENTENDI.**

Además de PID y PPID, los procesos en GNU/Linux también tienen otros atributos, como:

- **\*\*Estado del Proceso:\*\*** Puede ser "Ejecución", "Listo", "Espera" u "Otros".
- **\*\*Prioridad:\*\*** Nivel de prioridad del proceso en relación con otros procesos.
- **\*\*Uso de CPU y Memoria:\*\*** Información sobre el uso actual de la CPU y la memoria por parte del proceso.
- **\*\*Permisos:\*\*** Permisos de acceso del proceso a recursos del sistema.
- **\*\*Usuario y Grupo:\*\*** El usuario y el grupo al que pertenece el proceso.
- **\*\*Contexto de Archivos:\*\*** Los archivos que el proceso tiene abiertos y su estado.
- **\*\*Tiempo de Ejecución:\*\*** El tiempo que el proceso ha estado en ejecución.
- **\*\*Directorio de Trabajo Actual:\*\*** El directorio en el que el proceso está trabajando actualmente.
- **\*\*Estado de Espera:\*\*** Si el proceso está en espera de algún evento o recurso.

Estos atributos permiten a los sistemas operativos, como GNU/Linux, administrar eficazmente la ejecución de procesos, garantizar la seguridad y el aislamiento, y gestionar los recursos del sistema de manera adecuada.

b)

Para ver qué procesos están en ejecución en un sistema GNU/Linux, puedes utilizar varios comandos. Aquí tienes algunos de los comandos más comunes:

1. **\*\*`ps`:\*\*** El comando `ps` (Process Status) muestra información sobre los procesos en ejecución. Puedes usar diferentes opciones para personalizar la salida. Algunos ejemplos son:

- **`ps aux`:** Muestra una lista completa de todos los procesos en ejecución.
- **`ps aux | grep proceso`:** Filtra la salida para mostrar solo los procesos que coincidan con "proceso".

- `ps -e`: Muestra una lista simple de procesos activos.

2. `top`: El comando `top` muestra una lista en tiempo real de los procesos en ejecución y su uso de recursos, como la CPU y la memoria. Es interactivo y muestra una actualización continua.

4. `pgrep`: El comando `pgrep` se usa para buscar y mostrar los PIDs de los procesos que coinciden con un patrón específico. Por ejemplo: `pgrep proceso`.

5. `pstree`: Muestra los procesos en forma de árbol, lo que puede ayudar a visualizar la jerarquía de procesos.

6. `ps auxf`: Muestra una vista jerárquica de los procesos, mostrando cómo están relacionados entre sí.

7. `ps -eH`: Muestra los procesos en una estructura de árbol jerárquica.

Estos comandos te brindarán información sobre los procesos en ejecución en tu sistema. Puedes ajustar las opciones según tus necesidades para obtener la información específica que estás buscando.

c)

Cuando se habla de procesos en sistemas operativos Unix-like, como GNU/Linux, "background" y "foreground" se refieren a cómo se ejecuta un proceso en relación con la interacción del usuario con la terminal.

- **Background (Segundo Plano)**: Un proceso se ejecuta en segundo plano cuando no bloquea la terminal y permite que el usuario continúe interactuando con ella. En otras palabras, el proceso se ejecuta en segundo plano sin requerir la atención activa del usuario. Puedes ejecutar un proceso en segundo plano agregando el símbolo "&" al final del comando en la terminal. Por ejemplo:

```
...  
comando & # Ejecuta "comando" en segundo plano  
...
```

- **Foreground (Primer Plano)**: Un proceso se ejecuta en primer plano cuando bloquea la terminal y requiere la atención y entrada del usuario. El proceso en primer plano se ejecuta interactivamente y el usuario puede interactuar directamente con él, proporcionando entradas y viendo sus salidas. Por defecto, cuando ejecutas un comando en la terminal sin "&" al final, el proceso se ejecuta en primer plano.

La ejecución en primer plano y en segundo plano es importante cuando tienes tareas que pueden tomar tiempo pero no necesitas interactuar directamente con ellas todo el tiempo. Al ejecutar procesos en segundo plano, puedes continuar trabajando con la terminal mientras el proceso se ejecuta en el fondo. En cambio, cuando ejecutas un proceso en primer plano, este requiere tu atención y puede bloquear la terminal hasta que se complete.

d)

Para cambiar un proceso entre el primer plano (foreground) y el segundo plano (background) en un sistema Linux con Debian, puedes usar los siguientes comandos y atajos de teclado:

1. **\*\*Para pasar un proceso al primer plano (foreground):\*\***

Si tienes un proceso en segundo plano y deseas traerlo al primer plano para interactuar con él:

- Utiliza el comando ``fg`` seguido por el número de trabajo del proceso (proporcionado cuando suspendes el proceso con ``Ctrl + Z``).

Ejemplo:

...

```
fg %1
```

...

Esto traerá el proceso con el número de trabajo 1 al primer plano.

2. **\*\*Para pasar un proceso al segundo plano (background):\*\***

Si tienes un proceso en primer plano y deseas pasarlo al segundo plano:

- Primero, suspende el proceso en primer plano con ``Ctrl + Z``.
- Luego, usa el comando ``bg`` para reanudar el proceso en segundo plano.

Ejemplo:

...

```
bg %1
```

...

Esto reanudará el proceso con el número de trabajo 1 en segundo plano.

Recuerda que el número de trabajo se refiere al número asignado al proceso en el historial de trabajos de la terminal. Puedes ver una lista de procesos en segundo plano y primer plano utilizando el comando ``jobs``.

Estos comandos y atajos te permiten administrar los procesos en el primer plano y el segundo plano de manera eficiente mientras trabajas en la terminal.

- e) El `"|"` nos permite comunicar dos procesos por medio de un pipe o tubería desde la shell
- ^ El pipe conecta stdout (salida estándar) del primer comando con la stdin (entrada estándar) del segundo
- Básicamente se ejecuta el primer comando y la salida del mismo es enviada como entrada del segundo (o del siguiente a él) comando, ej: `$ ls | more`.

- f) **Redirecciones:** Puedes redirigir la salida estándar de la shell a un archivo. La salida estándar es lo que se imprime en pantalla. Por lo tanto en vez de escribir en pantalla, vas a poder escribir en un archivo, osea, lo que se te debería haber mostrado en la pantalla no lo hará y aparecerá dentro de ese archivo que le indicaste. Esta manera puede ser de manera **destructiva** (cuando redireccionas algo a un archivo, ese archivo (si existe) lo borra y crea uno nuevo con ese contenido (si no existía lo crea y le carga el contenido) o **no destructiva** (si ese archivo existe, se agrega la información al final, y si no existe lo crea y le carga la info).

g)

El comando ``kill`` en Linux se utiliza para enviar señales a procesos en ejecución. Aunque el nombre puede parecer un poco engañoso, su función principal no es necesariamente matar un proceso de inmediato, sino comunicarse con él y solicitar que realice una acción específica en función de la señal enviada.

La señal más común enviada por el comando ``kill`` es la señal SIGTERM (15), que solicita amablemente al proceso que se termine de manera ordenada. Si un proceso no responde a esta señal, también se puede enviar la señal SIGKILL (9), que es más agresiva y fuerza al proceso a finalizar inmediatamente sin importar su estado.

Aquí tienes ejemplos de uso del comando ``kill``:

1. **\*\*Terminar un proceso específico usando PID:\*\***

```
...  
kill PID  
...
```

Donde "PID" es el número de identificación del proceso que deseas terminar.

2. **\*\*Terminar un proceso de manera forzada usando PID:\*\***

```
...  
kill -9 PID  
...
```

Esto envía la señal SIGKILL para terminar el proceso inmediatamente.

3. **\*\*Terminar un proceso por su nombre usando ``pkill``:**

```
...  
pkill proceso  
...
```

Donde "proceso" es el nombre del proceso que deseas terminar.

4. **\*\*Enviar una señal específica a un proceso usando su PID:\*\***

```
...  
kill -SIGNUM PID  
...
```

Donde "SIGNUM" es el número de la señal que deseas enviar y "PID" es el número de identificación del proceso.

5. **\*\*Terminar todos los procesos de un usuario:\*\***

```
...  
pkill -u nombre_usuario  
...
```

Esto enviará la señal SIGTERM a todos los procesos del usuario "nombre\_usuario".

Recuerda que, en general, es preferible intentar terminar un proceso de manera ordenada usando `kill` con SIGTERM antes de recurrir a SIGKILL.

h)

Aquí tienes información sobre los comandos relacionados con el manejo de procesos en GNU/Linux y una comparación entre ellos:

1. **\*\*`ps`:\*\*** El comando `ps` (Process Status) se utiliza para mostrar información sobre los procesos en ejecución en el sistema. Puedes usar diversas opciones para personalizar la salida y mostrar detalles como el estado, uso de recursos, propietario y más.

Ejemplo:  
...  
ps aux  
...

2. **\*\*`kill`:\*\*** El comando `kill` se utiliza para enviar señales a procesos en ejecución. Se puede utilizar para finalizar procesos o solicitar que realicen ciertas acciones, como reiniciar o recargar.

Ejemplo:  
...  
kill PID  
...

3. **\*\*`pstree`:\*\*** El comando `pstree` muestra una representación jerárquica de los procesos en forma de árbol, lo que facilita la comprensión de cómo están relacionados.

Ejemplo:  
...  
pstree  
...

4. **killall**: El comando `killall` se utiliza para enviar señales a procesos según su nombre en lugar de su PID. Puedes finalizar todos los procesos con un nombre específico.

Ejemplo:

...

`killall proceso`

...

5. **top**: El comando `top` muestra una lista en tiempo real de los procesos en ejecución y su uso de recursos. Es una herramienta interactiva que proporciona una visión general de los procesos y su rendimiento.

Ejemplo:

...

`top`

...

6. **nice**: El comando `nice` se utiliza para establecer la prioridad de ejecución de un proceso. Puedes usarlo para ajustar la prioridad de CPU de un proceso en ejecución o al iniciar uno nuevo.

Ejemplo:

...

`nice -n 10 comando`

...

Comparación:

- `ps` muestra información sobre los procesos en ejecución.
- `kill` envía señales a procesos, como terminarlos.
- `pstree` muestra la jerarquía de procesos en forma de árbol.
- `killall` envía señales a procesos según su nombre.
- `top` proporciona una vista en tiempo real de los procesos y su uso de recursos.
- `nice` ajusta la prioridad de CPU de un proceso.

Cada comando tiene su propia función específica en el manejo y monitoreo de procesos en GNU/Linux.

6) a) El concepto de "empaquetar archivos" en GNU/Linux se refiere a la acción de combinar múltiples archivos y/o directorios en un solo archivo, conocido como un "paquete". Estos paquetes a menudo se utilizan para distribuir software, archivos de configuración, recursos y otros contenidos de manera organizada y conveniente.

b) Comandos realizados:

1. Primero creo los archivos.
2. Cargo valores a cada uno.
3. Calculo el tamaño total de esos archivos con el comando `du -c archivos`.



```
luciano@Debian:~$ du -c archivo1.txt archivo2.txt archivo3.txt prueba.exe
4      archivo1.txt
4      archivo2.txt
4      archivo3.txt
4      prueba.exe
16     total
```

4. Luego realizo el empaquetado con el comando “tar -cvf nombre\_del\_archivo.tar archivos\_directorios”:

```
luciano@Debian:~$ tar -cvf empaquetado.tar archivo1.txt archivo2.txt archivo3.tx
t prueba.exe
archivo1.txt
archivo2.txt
archivo3.txt
prueba.exe
```

al hacer el comando “du” que me permite saber el tamaño de uno o varios archivos juntos se observa que el archivo empaquetado ocupa 12k y el conjunto de archivos creados al principio ocupan 16k.

### El comando “DU”: (PARA VER EL TAMAÑO DE UN ARCHIVO)

Para ver cuánto pesa un archivo en Linux Debian, puedes utilizar el comando du (disk usage) con la opción -h (human-readable) para mostrar el tamaño en un formato más legible por humanos: “du -h nombre\_del\_archivo”.

c) (SE REFIERE A LOS COMANDOS QUE DEBO REALIZAR YO COMO

USUARIO?): SE REALIZA ESTE COMANDO: “ tar -cvf archivos\_empaquetados.tar archivo1.txt archivo2.txt”.

d) Si, con este comando “tar -cvf archivos\_empaquetados.tar archivo1.txt archivo2.txt.

### e) Descripción de los comandos:

1. **\*\*tar:\*\*** El comando `tar` se utiliza para empaquetar y desempaquetar archivos y directorios en un archivo. También puede aplicar compresión en el proceso. Es una herramienta muy útil para la creación de copias de seguridad y la transferencia de archivos.

Ejemplo de empaquetar y comprimir un directorio:

```
``bash
tar -czvf archivos.tar.gz directorio/
``
```

2. **\*\*grep:\*\*** El comando `grep` se utiliza para buscar patrones de texto en archivos o en la salida de otros comandos. Es una herramienta poderosa para filtrar y encontrar contenido específico en archivos.

- Buscar todos los archivos que contengan la cadena **pepe** en el directorio `/tmp`

```
grep pepe /tmp/*
```

Otro ejemplo:

```
luciano@Debian:~/practica3$ grep if /home/luciano/practica3/*
grep: /home/luciano/practica3/holanda: Is a directory
grep: /home/luciano/practica3/prueba14: Is a directory
grep: /home/luciano/practica3/punto30: Is a directory
/home/luciano/practica3/script12ab.sh:if [ $# -ne 2 ]; then #verifico que se hayan proporcionado dos parametros
/home/luciano/practica3/script12ab.sh:if [ $1 -gt $2 ]; then
/home/luciano/practica3/script12ab.sh:elif [ $1 -eq $2 ]; then
/home/luciano/practica3/script12c.sh:if [ $# -ne 3 ]; then
/home/luciano/practica3/script13c.sh:if [ $# -ne 1 ]; then
/home/luciano/practica3/script13c.sh:if [ -e $1 ]; then
/home/luciano/practica3/script13c.sh:if [ -d $1 ]; then
/home/luciano/practica3/script13c.sh:elif [ -f $1 ]; then
/home/luciano/practica3/script14.sh:# verifico que se haya ingresado algo como parametro
/home/luciano/practica3/script14.sh:if [ $# -ne 3 ]; then
/home/luciano/practica3/script14.sh:#verifico que sea directorio
/home/luciano/practica3/script14.sh:if [ ! -d $directorio ]; then
/home/luciano/practica3/script14.sh:#verifico que sea una opcion valida
/home/luciano/practica3/script14.sh:if [ $3 = "a" ] || [ $3 = "b" ]; then
/home/luciano/practica3/script14.sh:if [ -f $archivo ]; then
/home/luciano/practica3/script14.sh:if [ $opcion = "a" ]; then
/home/luciano/practica3/script16.sh:#Verificar si se proporcionó la extensión como parámetro
/home/luciano/practica3/script16.sh:if [ $# -ne 1 ]; then
/home/luciano/practica3/script18.sh:if [ $# -ne 1 ]; then
/home/luciano/practica3/script18.sh:if who | grep -q $1; then #el -q lo que te permite es que no se muestre nada en la pantalla al momento de efectuar ese comando
/home/luciano/practica3/script20.sh:if [ $# -ne 0 ]; then
/home/luciano/practica3/script20.sh:if [ ${#pila[@]} -eq 0 ]; then
/home/luciano/practica3/script23.sh:if [ $(($v % 2)) -eq 0 ]; then
/home/luciano/practica3/script25.sh:if [ $# -ne 1 ] && [ $# -ne 2 ]; then
/home/luciano/practica3/script25.sh:#verifico si el primer parametro es un string
/home/luciano/practica3/script25.sh:if [[ -n "$1" && ! "$1" =~ [0-9] ]]; then #se usa doble corchete ya que asi se evaluan expresiones condicionales, si la expresion es verdadera devuelve 0 y sino otro valor
V devuelve 0 y sino otro valor
```

**Ojo que si pones que busque en un directorio NO TE LO TOMA solamente se hace sobre archivos, no directorios. EN FIND SI TE DEJA, ES UNA DIFERENCIA.**

**Además fijate que te imprime toda la linea en donde se encuentra eso que andas buscando.**

**Si no quieres que no devuelva nada pones por ej `grep -q if /home/luciano/practica3/*` (punto 18 practica 3 hay uno que lo tengo que usar)**

Ejemplo de buscar una palabra en un archivo:

```
``bash
grep "palabra" archivo.txt
``
```

**3. `**gzip**` (podes tener un archivo `.tar.gz`) El comando ``gzip`` se utiliza para comprimir archivos. Crea un archivo comprimido con la extensión `".gz"` y **reduce el tamaño del archivo original**.**

Ejemplo de comprimir un archivo:

```
``bash
gzip archivo.txt
``
```

**`gzip -d archivo.tar.gz # Descomprime archivo.tar`**

4. **\*\*zgrep:\*\*** El comando `zgrep` es similar a `grep`, pero está diseñado para buscar patrones en archivos comprimidos (.gz). Permite buscar en archivos comprimidos sin necesidad de descomprimirlos previamente.

Ejemplo de buscar una palabra en un archivo comprimido:

```
``bash
zgrep "palabra" archivo.txt.gz
``
```

5. **\*\*wc:\*\*** El comando `wc` (word count) se utiliza para contar el número de líneas, palabras y caracteres en un archivo o en la entrada estándar. Es útil para obtener estadísticas sobre el contenido de un archivo.

Ejemplo de contar líneas, palabras y caracteres en un archivo:

```
``bash
wc archivo.txt
``
```

Estos comandos son herramientas fundamentales en la línea de comandos de Linux y son ampliamente utilizados para diversas tareas relacionadas con el manejo y análisis de archivos y datos.

7) `ls -l > prueba` carga en el archivo prueba (SI NO EXISTE TE LO CREA) la información de todos los archivos que se encuentran en el directorio principal del usuario:

```

-rw-r--r-- 1 luciano luciano    57 Aug 31 21:39 archivo1.txt
-rw-r--r-- 1 luciano luciano    53 Aug 31 21:41 archivo2.txt
-rw-r--r-- 1 luciano luciano    49 Aug 31 21:41 archivo3.txt
drwxr-xr-x 2 luciano luciano 4096 Aug 28 18:41 Desktop
drwxr-xr-x 2 luciano luciano 4096 Aug 28 18:41 Documents
drwxr-xr-x 2 luciano luciano 4096 Aug 28 18:41 Downloads
-rw-r--r-- 1 luciano luciano 10240 Aug 31 21:41 empaquetado.tar
drwxr-xr-x 2 luciano luciano 4096 Aug 28 18:41 Music
drwxr-xr-x 2 luciano luciano 4096 Aug 28 18:41 Pictures
-rw-r--r-- 1 luciano luciano     0 Sep  1 18:04 prueba
-rw-r--r-- 1 luciano luciano    44 Aug 28 19:04 prueba.exe
drwxr-xr-x 2 luciano luciano 4096 Aug 28 18:41 Public
drwxr-xr-x 2 luciano luciano 4096 Aug 28 18:41 Templates
drwxr-xr-x 2 luciano luciano 4096 Aug 28 18:41 Videos
~
~
~
~
~
~
```

**ps > PRUEBA:** Este comando muestra una lista de procesos en ejecución y redirige la salida al archivo "PRUEBA". Sin embargo, como el usuario no es root, es posible que no pueda acceder a todos los procesos, y algunos detalles pueden quedar ocultos.

**chmod 710 prueba:** Este comando cambia los permisos del archivo "prueba" para que el propietario tenga permisos de lectura, escritura y ejecución, el grupo tenga permisos de ejecución, pero otros usuarios no tengan permisos.

**chown root:root PRUEBA:** Este comando cambia el propietario y el grupo del archivo "PRUEBA" a "root" y "root", respectivamente. **Sin embargo, si el usuario que lo ejecuta no es root ni pertenece al grupo root, no tiene permisos para cambiar el propietario del archivo:**

```
luciano@Debian:~$ chown root:root PRUEBA
chown: changing ownership of 'PRUEBA': Operation not permitted
luciano@Debian:~$
```

**chmod 777 PRUEBA:** les da permisos a todos los usuarios (lectura, escritura y ejecución) sobre el archivo "PRUEBA".

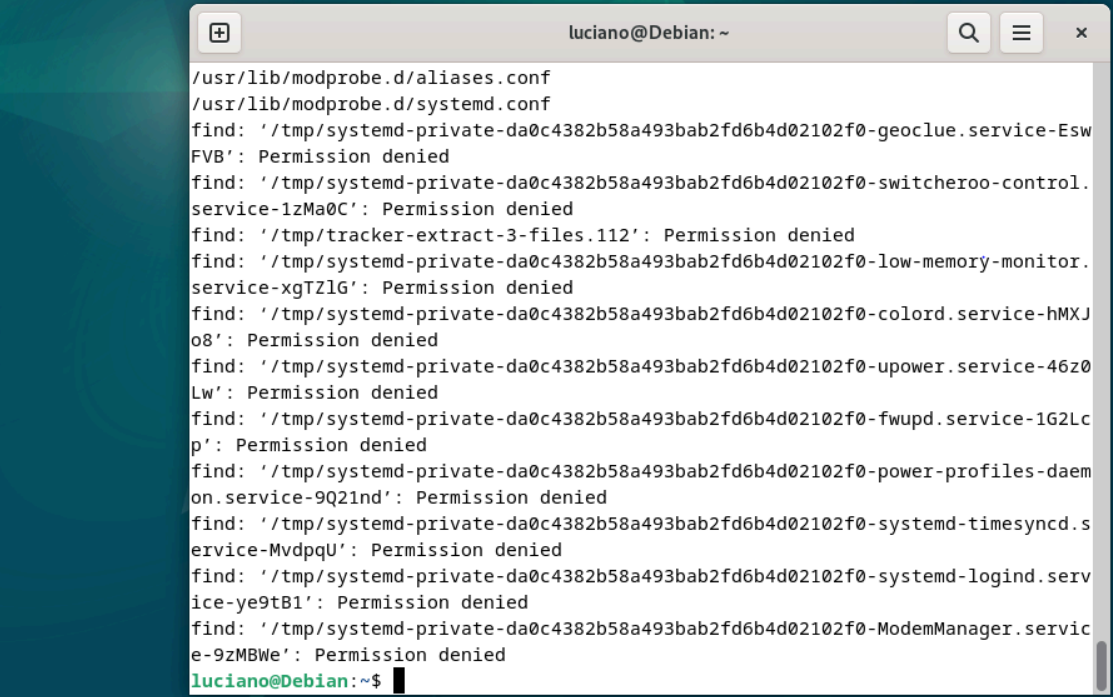
**chmod 700 /etc/passwd:** Este comando cambia los permisos del archivo "/etc/passwd" para que solo el propietario tenga acceso de lectura, escritura y ejecución. Esto es importante ya que "/etc/passwd" contiene información importante de los usuarios del sistema.

**passwd root:** Este comando intenta cambiar la contraseña del usuario "root". Si el usuario que lo ejecuta no tiene privilegios de root, no podrá cambiar la contraseña.

**rm PRUEBA:** borra el archivo.

**man /etc/shadow:** Este comando intenta mostrar el manual del archivo "/etc/shadow". Sin embargo, dado que "/etc/shadow" contiene información confidencial de contraseñas y solo es accesible por root.

**find / -name \*.conf:** Este comando busca archivos con extensión ".conf" en todo el sistema de archivos ("/"). Puede mostrar resultados si el usuario tiene permisos para acceder a los directorios:



```
luciano@Debian: ~  
/usr/lib/modprobe.d/aliases.conf  
/usr/lib/modprobe.d/systemd.conf  
find: '/tmp/systemd-private-da0c4382b58a493bab2fd6b4d02102f0-geoclue.service-Esw  
FVB': Permission denied  
find: '/tmp/systemd-private-da0c4382b58a493bab2fd6b4d02102f0-switcheroo-control.  
service-1zMa0C': Permission denied  
find: '/tmp/tracker-extract-3-files.112': Permission denied  
find: '/tmp/systemd-private-da0c4382b58a493bab2fd6b4d02102f0-low-memory-monitor.  
service-xgTZLG': Permission denied  
find: '/tmp/systemd-private-da0c4382b58a493bab2fd6b4d02102f0-colord.service-hMXJ  
o8': Permission denied  
find: '/tmp/systemd-private-da0c4382b58a493bab2fd6b4d02102f0-upower.service-46z0  
Lw': Permission denied  
find: '/tmp/systemd-private-da0c4382b58a493bab2fd6b4d02102f0-fwupd.service-1G2Lc  
p': Permission denied  
find: '/tmp/systemd-private-da0c4382b58a493bab2fd6b4d02102f0-power-profiles-daem  
on.service-9Q21nd': Permission denied  
find: '/tmp/systemd-private-da0c4382b58a493bab2fd6b4d02102f0-systemd-timesyncd.s  
ervice-MvdpqU': Permission denied  
find: '/tmp/systemd-private-da0c4382b58a493bab2fd6b4d02102f0-systemd-logind.serv  
ice-ye9tB1': Permission denied  
find: '/tmp/systemd-private-da0c4382b58a493bab2fd6b4d02102f0-ModemManager.servic  
e-9zMBWe': Permission denied  
luciano@Debian:~$
```

**usermod root -d /home/newroot -L:** Este comando intenta cambiar el directorio de inicio del usuario "root" a "/home/newroot". Sin embargo, solo root tiene permisos para modificar la configuración del usuario "root".

**cd /root:** Este comando intenta cambiar al directorio "/root", que es el directorio de inicio del usuario "root". Si el usuario que lo ejecuta no es root, no tendrá acceso a este directorio.

**rm \*:** Este comando intenta eliminar todos los archivos en el directorio actual. Sin embargo, si el usuario no tiene permisos para eliminar algunos archivos, podría mostrar mensajes de error.

**cd /etc:** Este comando cambia al directorio "/etc", que contiene archivos de configuración del sistema

**cp \* /home -R:** Este comando copia todos los archivos del directorio "/etc" al directorio "/home" de forma recursiva ("-R"). Sin embargo, si el usuario no es root, es posible que no tenga permisos para acceder a todos los archivos en "/etc".

**shutdown:** Este comando apaga o reinicia el sistema, pero solo root tiene permisos para ejecutarlo. El usuario no root probablemente no podrá realizar un apagado o reinicio del sistema.

8)

a) kill 23

b) killall init. Si terminas el proceso del init, el sistema operativo dejará de funcionar y es muy probable que te desconectes de la sesión en curso. El sistema operativo se bloqueará

y se volverá inoperable. Esto puede causar la pérdida de datos y puede requerir un reinicio forzado del sistema para recuperarlo.

**c)** `find /home -type f -name ".conf"` (no debe haber ninguna pq no me funca asi, pero si aumento el rango de busqueda asi: `sudo find / -type f -name ".conf"` si me busca).

- `sudo`: El comando `sudo` se utiliza para ejecutar el comando como superusuario o con privilegios elevados. Esto es necesario porque, en muchos sistemas, algunos directorios en `/home` pueden tener restricciones de acceso para usuarios normales.
- `find`: `find` es el comando utilizado para buscar archivos y directorios en el sistema de archivos.
- `/home`: Este es el directorio principal donde se encuentran los directorios de usuarios en muchos sistemas Linux. El comando `find` comenzará a buscar archivos desde este directorio.
- `-type f`: Este es un argumento de `find` que especifica que solo se deben buscar archivos regulares (no directorios ni otros tipos de archivos). Esto es útil para asegurarse de que solo se muestren archivos y no directorios en los resultados.
- `-name "*\.conf"`: Este es otro argumento de `find` que establece el patrón de búsqueda. En este caso, busca archivos cuyos nombres coincidan con el patrón `*\.conf`. Aquí, el `*` actúa como comodín, coincidiendo con cualquier cadena de caracteres, y `\.` se utiliza para buscar archivos con la extensión `.conf`. El `\` se utiliza para escapar el punto (`.`) ya que el punto es un carácter especial en las expresiones regulares.

d) `ps aux > /home/<su_nombre_de_usuario>/procesos`

**e)** `chmod 731 /home/<su nombre de usuario>/xxx.`

**f)** `chmod 750 /home/<su nombre de usuario>/yyy.`

g)     -1 `cd /tmp`  
       2- `rm *`

ó: `rm /tmp/*`

El directorio `/tmp` (abreviatura de "temporary," que significa temporal en inglés) es un directorio en sistemas Linux y Unix diseñado para almacenar archivos temporales. En general, cualquier usuario o programa puede crear y escribir archivos temporales en este directorio. A continuación, se describe qué se encuentra comúnmente en el directorio `/tmp`:

1. **Archivos temporales:** `/tmp` almacena una variedad de archivos temporales que son utilizados por programas y procesos en ejecución. Estos archivos pueden incluir información

temporal, cachés, colas de correo, archivos de registro temporales y otros datos temporales generados por aplicaciones en el sistema.

2. **Sockets y pipes:** A menudo, se crean sockets (para comunicación interprocesos) y pipes (tuberías) temporales en `/tmp` para permitir la comunicación entre programas.
3. **Directorios temporales:** Algunos programas y aplicaciones crean directorios temporales dentro de `/tmp` para realizar tareas temporales o almacenar datos temporales.
4. **Archivos de bloqueo:** En ocasiones, se utilizan archivos de bloqueo en `/tmp` para evitar que múltiples instancias de un programa se ejecuten al mismo tiempo o para señalar eventos importantes en el sistema.
5. **Otros archivos y datos temporales:** La variedad de datos almacenados en `/tmp` puede ser amplia y depende de cómo se utilice el sistema y las aplicaciones instaladas en él.

Es importante destacar que los archivos en `/tmp` no suelen persistir entre reinicios del sistema y pueden ser eliminados automáticamente por el sistema operativo para liberar espacio. Por esta razón, los archivos importantes no deben almacenarse en `/tmp`, ya que podrían perderse.

Ten en cuenta que la limpieza de `/tmp` y la gestión de archivos temporales es responsabilidad del sistema operativo y de las aplicaciones que los crean. Los usuarios no deben borrar manualmente archivos en `/tmp`, a menos que estén seguros de que ya no se necesitan y que su eliminación no afectará el funcionamiento del sistema o las aplicaciones.

h) `chown iso2021 /opt/isodata`

i) `pwd > /home/<su_nombre_de_usuario>/donde`

9)

a) "su"

b) `sudo adduser Napellido` (siendo N la primera letra de tu nombre). Luego: `sudo passwd usuario`

c) Cuando creas un usuario en Debian Linux, se realizan varias modificaciones en el sistema de archivos y se crean varios directorios y archivos relacionados con el nuevo usuario. A continuación, se describen los cambios más importantes:

1. **Creación del directorio de inicio del usuario:** Cuando creas un usuario, se crea automáticamente un directorio de inicio en `/home` con el nombre de usuario. Por ejemplo, si el nombre de usuario es "NAellido", se creará un directorio `/home/NAellido`.
2. **Modificación del archivo `/etc/passwd`:** El archivo `/etc/passwd` contiene información sobre todos los usuarios del sistema. Después de crear un usuario, se añadirá una nueva entrada en este archivo para el usuario recién creado. La entrada contendrá información



como el nombre de usuario, el identificador de usuario (UID), el identificador de grupo (GID), el directorio de inicio y la shell predeterminada.

3. **Modificación del archivo `/etc/shadow`:** El archivo `/etc/shadow` almacena las contraseñas encriptadas de los usuarios. Después de asignar una contraseña al usuario, se actualizará este archivo con la información de la contraseña encriptada del nuevo usuario.

4. **Creación del directorio de inicio del usuario:** Como se mencionó anteriormente, se crea un directorio de inicio para el usuario en `/home`.

5. **Modificación de los archivos de grupo:** El usuario recién creado se agregará al grupo primario con el mismo nombre que el usuario. Por ejemplo, si el nombre de usuario es "NApellido", se creará un grupo con el nombre "NApellido" y se asignará como grupo primario del usuario. Las modificaciones relacionadas con grupos se registran en archivos como `/etc/group` y `/etc/gshadow`.

6. **Creación de archivos de configuración específicos del usuario:** En el directorio de inicio del usuario (`/home/NApellido` en este ejemplo), se crean archivos y directorios de configuración específicos para el usuario, como `.bashrc`, `.profile`, `.bash_history`, entre otros, dependiendo del entorno de shell y las preferencias del usuario.

7. **Otros archivos y directorios de configuración personalizada:** Además de los archivos mencionados anteriormente, pueden crearse otros archivos y directorios de configuración específicos para aplicaciones y servicios que el usuario utilice, como `.config`, `.local`, `.ssh` (para claves SSH) y otros.

d) `cd /temp`. Luego: `mkdir cursada2021`. Ó `mkdir /temp/cursada2021`.

e) Para copiar todos los archivos de `/var/log` al directorio `/tmp/cursada2021` que acabas de crear, puedes utilizar el comando `cp`, pero ten en cuenta que esto puede generar una gran cantidad de archivos, por lo que es importante tener cuidado. Aquí te explico cómo hacerlo:

1. Abre una terminal en Debian.

2. Utiliza el siguiente comando para copiar todos los archivos de `/var/log` al directorio `/tmp/cursada2021`:

```
```bash
sudo cp -r /var/log/* /tmp/cursada2021
```
```

Aquí hay una explicación de los componentes de este comando:

- `sudo`: Usamos `sudo` para asegurarnos de que tengamos los permisos necesarios para copiar archivos de `/var/log`, ya que algunos de estos archivos pueden requerir privilegios de superusuario.

- `cp`: Es el comando para copiar archivos y directorios.

- `-r`: Esta opción se utiliza para copiar directorios y su contenido de manera recursiva. Esto es importante ya que `/var/log` contiene varios subdirectorios y archivos anidados. **Si vos lo quieres hacer sin eso, y /var/log o el que fuese tiene directorios dentro, no te va a funcionar, te va a decir que le tenes que poner -r.**

- `/var/log/*`: Esto especifica que queremos copiar todos los archivos y directorios dentro de `/var/log`. El asterisco `*` se utiliza como comodín para seleccionar todo el contenido de `/var/log`.

- `/tmp/cursada2021`: Esta es la ubicación de destino donde copiaremos los archivos.

3. El comando copiará todos los archivos y directorios de `/var/log` al directorio `/tmp/cursada2021`. Dependiendo de la cantidad de archivos en `/var/log`, este proceso puede llevar tiempo.

4. Puedes verificar que los archivos se hayan copiado correctamente utilizando el comando `ls` para listar los contenidos de `/tmp/cursada2021`:

```
```bash
ls /tmp/cursada2021
```
```

Deberías ver una lista de archivos y directorios copiados desde `/var/log` en `/tmp/cursada2021`.

Es importante tener en cuenta que la copia de todos los archivos de `/var/log` ocupará espacio en disco y puede ser innecesaria, por lo que asegúrate de que esta acción sea realmente lo que deseas hacer. También ten en cuenta que algunos archivos de registro pueden contener información sensible o privada, así que ten cuidado al manejarlos.

f) `sudo chown Napellido:users /tmp/cursada2021`. (si quieres cambiar los dueños y grupos de todos los archivos que este directorio contenga adentro vas a tener que agregar luego de "chown" un "-r").

**g) `sudo chmod -r 723 /tmp/cursada2021`. (te cambia los permisos tanto del directorio como dentro de los archivos de ese directorio).**

**Si quieres excluir al directorio tenes que hacer esto:**

```
```bash
find /ruta/al/directorio -type f -exec chmod 711 {} \;
```
```

- `/ruta/al/directorio`: Reemplaza esto con la ruta completa al directorio en el que deseas cambiar los permisos de los archivos.

- `-type f`: Esta opción de `find` especifica que solo se deben aplicar cambios a los archivos (excluyendo directorios).

- **`exec chmod 711 {} \;`**: Esto ejecuta el comando **`chmod 711`** en cada archivo encontrado por **`find`**. Los corchetes **`{}`** se utilizan para representar cada archivo encontrado.

Este comando cambiará los permisos de todos los archivos dentro del directorio, de forma recursiva, sin afectar al propio directorio ni a sus subdirectorios.

Si deseas incluir subdirectorios y cambiar los permisos de los archivos en ellos, puedes eliminar la parte **`-type f`** del comando. En ese caso, el comando cambiará los permisos de todos los archivos y subdirectorios dentro del directorio especificado de forma recursiva.

h)

Para acceder a otra terminal virtual y luego iniciar sesión con el usuario que creaste previamente en Debian Linux, puedes seguir estos pasos:

1. Presiona las teclas **`Ctrl + Alt + F2`** para cambiar a otra terminal virtual. Puedes usar cualquier tecla de función de F1 a F6 para esto. Cada una de estas teclas representa una terminal virtual diferente.

2. Verás una pantalla de inicio de sesión en la nueva terminal virtual. Inicia sesión con el nombre de usuario y la contraseña que creaste previamente. Por ejemplo:

- Nombre de usuario: NApellido (reemplaza con el nombre de usuario que creaste).
- Contraseña: [Tu contraseña]

3. Después de ingresar el nombre de usuario y la contraseña, deberías tener acceso a la nueva terminal virtual con ese usuario.

4. Para cambiar entre las diferentes terminales virtuales puedes usar **`Ctrl + Alt + F1`** para volver a la terminal original en la que estabas trabajando o utilizar **`Ctrl + Alt + F3`**, **`Ctrl + Alt + F4`**, etc., para cambiar a otras terminales virtuales si están disponibles.

Recuerda que cada terminal virtual representa una sesión de usuario separada, por lo que puedes realizar tareas independientes en cada una de ellas. Si deseas cerrar una sesión de terminal virtual, puedes hacerlo normalmente con el comando **`exit`** o cerrando la ventana de la terminal.

i)

Puedes saber el nombre de la terminal en la que estás utilizando el comando **`tty`**. Este comando te mostrará el nombre de la terminal actual. Simplemente ejecuta:

```
```bash
tty
```
```

La terminal te devolverá algo como **`/dev/ttyX`**, donde **"X"** es un número que identifica la terminal actual. Por ejemplo:

```
...  
/dev/tty1  
...
```

Este número puede variar dependiendo de la terminal virtual en la que te encuentres. Si estás usando la terminal gráfica predeterminada (Ctrl + Alt + F1), generalmente será "/dev/tty1". Si estás en una terminal virtual diferente, el número cambiará en consecuencia.

j)

Para verificar la cantidad de procesos activos en el sistema en Linux, puedes utilizar el comando `ps` junto con el comando `wc`. Aquí te muestro cómo hacerlo:

```
``bash  
ps aux | wc -l  
...
```

Este comando cuenta la cantidad de líneas que se muestran al listar todos los procesos en ejecución con `ps aux`. **La opción `-l` de `wc`** se utiliza para contar líneas. Por lo tanto, el resultado te dará la cantidad total de procesos activos en el sistema.

Ten en cuenta que este número incluirá todos los procesos, tanto los del sistema como los de los usuarios. Si deseas ver una lista completa de procesos y su información detallada, puedes ejecutar simplemente:

```
``bash  
ps aux  
...
```

En la salida, verás información sobre cada proceso, incluyendo su identificador de proceso (PID), usuario que lo ejecuta, consumo de recursos y más. Puedes usar esta información para investigar procesos específicos en detalle.

k) Para verificar la cantidad de usuarios conectados al sistema en Linux, puedes utilizar varios comandos, dependiendo de cómo desees obtener esta información. Aquí te muestro algunos métodos:

**1. `w`:** El comando `w` muestra información sobre los usuarios conectados al sistema, incluyendo su nombre de usuario, terminal en la que están conectados, hora de inicio de sesión y más. Además, muestra la carga promedio del sistema. Ejecuta:

```
``bash  
w  
...
```

La columna "USER" muestra la lista de usuarios conectados.

2. **\*\*who:\*\*** El comando ``who`` muestra una lista de usuarios que han iniciado sesión en el sistema, junto con detalles como su nombre de usuario, terminal y hora de inicio de sesión. Ejecuta:

```
```bash
who
```
```

3. **\*\*users:\*\*** El comando ``users`` muestra simplemente una lista de los nombres de usuario de las personas que están actualmente conectadas al sistema. Ejecuta:

```
```bash
users
```
```

L)

Para enviar un mensaje al usuario creado desde la terminal del usuario root, puedes utilizar el comando ``write``. El comando ``write`` te permite enviar mensajes a otros usuarios que están conectados al sistema. Aquí tienes los pasos para hacerlo:

1. Cambia a la terminal del usuario root si aún no lo estás. Puedes abrir una nueva terminal y usar el comando ``su`` o ``sudo -i`` para obtener acceso a la cuenta de root, dependiendo de cómo esté configurado tu sistema.

2. Una vez que estés en la terminal de root, utiliza el comando ``write`` seguido del nombre de usuario al que deseas enviar el mensaje y luego presiona ``Enter``. Por ejemplo, si el nombre de usuario al que deseas enviar el mensaje es "NApellido", puedes usar:

```
```bash
write NApellido
```
```

Esto iniciará una sesión de mensajería con el usuario "NApellido".

3. A continuación, escribe tu mensaje y presiona ``Enter`` para enviarlo. Por ejemplo, puedes escribir:

```
```bash
El sistema se apagará en 10 minutos. Por favor, guarde su trabajo.
```
```

El usuario "NApellido" verá tu mensaje en su terminal.

4. Para finalizar la sesión de mensajería, presiona ``Ctrl + D`` o escribe ``/bye`` y presiona ``Enter``.

El usuario "NApellido" recibirá el mensaje que enviaste desde la terminal de root. Asegúrate de que el mensaje sea claro y contenga la información necesaria sobre el apagado del sistema.

m) shutdown.

10) a),b),c),d) lo hice en la compu facil. Para la última pregunta del D la razón en el cual puedes crear este archivo si ya existe el llamado "LEAME" es porque linux es case sensitive y además están en distintos directorios, uno está en el /home/luciano y el otro en /home/luciano/212767

e)

El comando `find` en Linux es una herramienta poderosa y versátil que se utiliza para buscar archivos y directorios en el sistema de archivos basándose en una variedad de criterios, como el nombre del archivo, la ubicación, el tamaño, la fecha de modificación y más. El comando `find` es una herramienta de línea de comandos extremadamente útil y flexible en sistemas Debian (y en otros sistemas basados en Unix) y es ampliamente utilizado para tareas de búsqueda y gestión de archivos.

**A continuación, te proporciono una descripción detallada de cómo utilizar el comando `find` en Linux Debian:**

**\*\*Sintaxis básica de `find`\*\*:**

```
```bash
```

```
find [ruta] [opciones] [expresión]
```

```
```
```

**PARTE A BUSCAR DE UNA RUTA QUE VOS LE INDICAS EN ADELANTE (DIGAMOS QUE RECURSIVAMENTE HASTA QUE NO HAYA MÁS DIRECTORIOS POR LAS CUALES BUSCAR), SI QUERES LIMITAR ESTO TENES QUE USAR EL `-maxdepth` y `-mindepth`).**

- **`ruta`**: Especifica el directorio base desde donde comenzará la búsqueda. Si no se proporciona, la búsqueda comenzará en el directorio actual.

- **`opciones`**: Son las opciones que personalizan el comportamiento de `find`. Algunas de las opciones comunes incluyen `-type`, `-name`, `-size`, `-mtime`, etc.

- **`expresión`**: Es la expresión que define los criterios de búsqueda.

**\*\*Ejemplos de uso\*\*:**

**1. \*\*Buscar por nombre de archivo o patrón de nombre\*\*:**

```
```bash
```

```
find /ruta -name "archivo.txt"
```

```
```
```

Este comando busca el archivo con el nombre "archivo.txt" en la ruta especificada.

## **2. \*\*Buscar por tipo de archivo (directorio, archivo, enlace simbólico, etc.)\*\*:**

```
```bash
find /ruta -type d
```
```

Este comando busca todos los directorios en la ruta especificada.

## **3. \*\*Buscar por tamaño de archivo\*\*:**

```
```bash
find /ruta -size +1M
```
```

Este comando busca archivos con un tamaño mayor a 1 megabyte en la ruta especificada.

## **4. \*\*Buscar por fecha de modificación\*\*:**

```
```bash
find /ruta -mtime -7
```
```

Este comando busca archivos modificados en los últimos 7 días en la ruta especificada.

## **5. \*\*Buscar y ejecutar acciones en archivos encontrados\*\*:**

```
```bash
find /ruta -name "*.txt" -exec cp {} /destino \;
```
```

Este comando busca todos los archivos con extensión ".txt" en la ruta especificada y los copia a un directorio de destino.

## **6. \*\*Buscar y excluir directorios o archivos específicos\*\*:**

```
```bash
find /ruta -type f -not -name "archivo.txt"
```
```

Este comando busca todos los archivos en la ruta especificada que no se llamen "archivo.txt".

## **7. \*\*Búsqueda recursiva en todo el sistema de archivos\*\*:**

```
```bash
```



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

Este comando buscará el archivo con el nombre "archivo.txt" en todo el sistema de archivos, comenzando desde la raíz ("/").

**\*\*Notas adicionales\*\***:

- `find` realiza búsquedas de forma recursiva por defecto, lo que significa que busca en todos los subdirectorios dentro de la ruta especificada a menos que se especifique lo contrario.
- `find` puede combinarse con otros comandos, como `grep`, `exec`, `xargs`, para realizar acciones específicas en los archivos encontrados.

## **explicación de la opciones que hay:**

El comando `find` en Linux ofrece una amplia gama de opciones que puedes usar para personalizar tus búsquedas de archivos y directorios. A continuación, te proporciono una descripción de algunas de las opciones más comunes:

1. **\*\*`-name`\*\***: Esta opción se utiliza para buscar archivos y directorios por su nombre. Puedes especificar un patrón de nombre, por ejemplo:

```
```bash
find /ruta -name "archivo.txt"
...
```

2. **\*\*`-type`\*\***: Permite buscar por tipo de archivo. Las opciones comunes incluyen:

- `d`: Directorio.
- `f`: Archivo regular.
- `l`: Enlace simbólico.
- `b`: Archivo de dispositivo de bloque.
- `c`: Archivo de dispositivo de caracteres.
- `p`: Tubería con nombre.
- `s`: Socket.

Ejemplo:

```
```bash
find /ruta -type d
...
```

3. **\*\*`-size`\*\***: Busca archivos por su tamaño. Puedes usar sufijos como `k` (kilobytes), `M` (megabytes), `G` (gigabytes) para especificar el tamaño. Ejemplo:

```
```bash
find /ruta -size +1M
```
```

4. **\*\*`-mtime`\*\***: Permite buscar archivos basados en su fecha de modificación. Puedes usar un número para indicar los días desde la última modificación. Para buscar archivos modificados en los últimos 7 días:

```
```bash
find /ruta -mtime -7
```
```

5. **\*\*`-exec`\*\***: Esta opción se utiliza para ejecutar un comando en los archivos encontrados. Debes usar `{}` como marcador de posición para los archivos encontrados y terminar el comando con `;`. Ejemplo:

```
```bash
find /ruta -name "*.txt" -exec cp {} /destino \;
```
```

6. **\*\*`-not`\*\***: Permite negar una condición. Por ejemplo, para buscar todos los archivos que no se llaman "archivo.txt":

```
```bash
find /ruta -type f -not -name "archivo.txt"
```
```

7. **\*\*`-maxdepth` y `-mindepth`\*\***: Estas opciones limitan la profundidad de búsqueda en directorios anidados. Por ejemplo, para buscar solo en el directorio actual y sus subdirectorios inmediatos:

```
```bash
find /ruta -maxdepth 1 -name "archivo.txt"
```
```

**-maxdepth 1**: Esta opción limita la búsqueda a un máximo de 1 nivel de profundidad. Esto significa que solo se buscarán archivos dentro de "/ruta" y no se buscarán archivos dentro de subdirectorios de "/ruta". Si deseas buscar en subdirectorios, puedes aumentar el valor después de "-maxdepth" (por ejemplo, "-maxdepth 2" para buscar hasta 2 niveles de profundidad).

El comando `find` se utiliza en sistemas Unix y Linux para buscar archivos y directorios en un sistema de archivos. La opción `-type` se usa para especificar el tipo de archivo que deseas buscar. Cuando usas **`find -type l`**, estás buscando enlaces simbólicos en el sistema de archivos.

Un enlace simbólico (también conocido como symlink) es un tipo especial de archivo que actúa como un acceso directo a otro archivo o directorio en el sistema de archivos. No contiene los datos del archivo real, sino que apunta al archivo o directorio al que se hace

referencia. Los enlaces simbólicos se utilizan para crear rutas o accesos más convenientes a archivos o directorios y para facilitar la administración de archivos.

Cuando ejecutas `find -type l`, el comando buscará en todo el sistema de archivos y mostrará una lista de todos los enlaces simbólicos que encuentre. Esto puede ser útil cuando deseas identificar enlaces simbólicos en tu sistema o cuando necesitas realizar operaciones específicas en ellos, como verificar si un enlace simbólico está roto o redirige a un archivo o directorio válido.

f) `find /ruta_del_directorio -type f -name "*.so" > /ruta_del_directorio/.ejercicio_f`  
**“EL “.” EN EL NOMBRE DEL ARCHIVO LO HACE OCULTO, PODES VERLO CON EL COMANDO `ls -a`.**

11)

**`mkdir iso`**= crea directorio.

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

Este comando se divide en dos partes:

1. `cd ./iso`: Cambia el directorio actual (`cd`) al subdirectorio llamado "iso" en el directorio de trabajo actual. Esto significa que te moverás al directorio "iso" dentro del directorio en el que te encuentras.

2. `ps > f0`: Una vez que estás en el directorio "iso", ejecuta el comando `ps` para listar los procesos en ejecución. Luego, redirige la salida de este comando (`>`) al archivo llamado "f0" en el directorio "iso".

Entonces, en resumen, el segundo comando primero te mueve al directorio "iso" y luego lista los procesos en ejecución y guarda esa lista en el archivo "f0" dentro del directorio "iso".

**`ls > f1`**: En el directorio "iso", ejecuta el comando `ls` y redirige su salida a un archivo llamado "f1". Esto también se ejecutará sin problemas.

**`cd /`**: Cambia al directorio raíz del sistema.

**`echo &home`**:

Este comando muestra en la terminal el contenido de la variable de entorno `$HOME`. La variable `$HOME` contiene la ubicación del directorio de inicio del usuario actual. Cuando ejecutas este comando, verás en la pantalla la ruta completa al directorio de inicio del usuario.

Por ejemplo, si la salida de `echo $HOME` es:

```
...  
/home/usuario  
...
```

Significaría que el directorio de inicio del usuario actual es "/home/usuario". Esta información es útil para saber cuál es el directorio de inicio del usuario y para realizar operaciones que involucren rutas absolutas en el sistema de archivos.

### **echo mejor explicado:**

El comando `echo` en Linux (y en otros sistemas Unix) se utiliza para mostrar mensajes de texto o variables en la terminal. Es una herramienta simple pero útil para imprimir texto en la pantalla. La sintaxis básica del comando `echo` es la siguiente:

```
``bash
echo [texto o variables]
``
```

- `[texto o variables]`: Puedes proporcionar un mensaje de texto entre comillas o variables que desees mostrar en la pantalla. Si usas variables, el comando imprimirá el valor de la variable.

Por ejemplo, aquí hay algunos ejemplos de uso del comando `echo`:

#### 1. Mostrar un mensaje de texto en la terminal:

```
``bash
echo "Hola, mundo!"
``
```

Esto mostrará "Hola, mundo!" en la terminal.

#### 2. Mostrar el valor de una variable:

```
``bash
nombre="Juan"
echo "Hola, $nombre"
``
```

Si la variable "nombre" contiene "Juan", esto mostrará "Hola, Juan" en la terminal.

#### 3. Mostrar el contenido de una variable de entorno:

```
``bash
echo $HOME
``
```

Esto mostrará la ubicación del directorio de inicio del usuario actual, ya que `$HOME` es una variable de entorno que almacena esta información.

#### 4. Mostrar múltiples mensajes:

```
```bash
echo "Este es un mensaje" "y este es otro mensaje"
```
```

Esto mostrará ambos mensajes en la misma línea separados por un espacio.

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

ls -l &> \$HOME/iso/ls: En el directorio raíz, ejecuta el comando ls -l y redirige tanto la salida estándar como la salida de error al archivo "ls" en el directorio "iso" dentro del directorio de inicio del usuario.

El símbolo "&" en el contexto del comando `ls -l &> \$HOME/iso/ls` se utiliza para redirigir tanto la salida estándar (STDOUT) como la salida de error (STDERR) del comando `ls -l` hacia un archivo. Veamos en detalle cómo funciona:

- `&>`: Este es un operador de redirección que combina tanto la salida estándar como la salida de error en una sola dirección, que en este caso es hacia un archivo.

- `\$HOME/iso/ls`: Esto es la ubicación del archivo de destino donde se redirigirán las salidas. `\$HOME` es una variable de entorno que generalmente representa el directorio de inicio del usuario, y `/iso/ls` es la ruta al archivo "ls" dentro de un directorio llamado "iso" en el directorio de inicio del usuario.

Entonces, cuando se ejecuta `ls -l &> \$HOME/iso/ls`, el resultado es que:

- La salida estándar (STDOUT) del comando `ls -l`, que normalmente se mostraría en la pantalla, se redirige al archivo `\$HOME/iso/ls`.

- La salida de error (STDERR) del comando `ls -l`, que normalmente se usaría para mostrar mensajes de error en la pantalla en caso de problemas, también se redirige al mismo archivo `\$HOME/iso/ls`.

En resumen, el uso de "&>" permite redirigir tanto la salida estándar como la salida de error del comando hacia un archivo específico. Esto es útil para capturar tanto los resultados exitosos como los mensajes de error en un archivo para su posterior revisión.

**cd \$HOME; mkdir f2:** Cambia al directorio de inicio del usuario y crea un directorio llamado "f2" allí.

**ls -ld f2:** (me dijeron que esta bien)creo que esta mal, debería ser ls -ld f2, pero si es así al no existir el archivo "df2" te va a decir que el mismo no existe, y si era ls -ld f2 y como el archivo se encuentra en el mismo directorio donde está parado ahora no hace falta poner la ruta en donde se encuentra.

**chmod 341 f2:** Le cambias los permisos al archivo f2.

**touch dir:** creas una archivo llamado dir.

**cd f2:** vas al directorio f2.

**cd ~/iso:** Vas al directorio /home/usuario/iso.

**pwd > f3:** carga la ruta donde te encuentras en el archivo f3. (si no existe te lo crea, como es en este caso).

**ps | grep 'ps' | wc -l >> ../f2/f3:** Ejecuta el comando ps, filtra las líneas que contienen "ps" utilizando grep, cuenta las líneas con wc -l y agrega el resultado al archivo "f3" en el directorio "f2". (esta bien eso q marque?). Al "f3" lo crea no?

**chmod 700 ../f2 ; cd .. :** (esta bien me dijo) Debería ser **chmod 700 ../f2; cd ..** el cual así Cambia los permisos del directorio "f2" y luego regresa al directorio padre.

**find . -name etc/passwd:** Realiza una búsqueda de archivos llamados "etc/passwd" a partir del directorio actual.

**find / -name etc/passwd:** Realiza una búsqueda de archivos llamados "etc/passwd" desde el directorio raíz. Esto buscará en todo el sistema de archivos y podría arrojar errores de permisos.

**mkdir ejercicio5:** Crea un directorio llamado "ejercicio5" en el directorio de trabajo actual del usuario.

b) 19: **cp -r /home/usuario/f2 /tmp/cursada2021.**

20: **cp -r /home/usuario/iso /home/usuario/f2 /home/usuario/dir /tmp/cursada2021.** (esta bien así? pq puse el -r para que tmb me cargue la informacion de adentro).

12) a) (no lo probe todo en la maquina virtual, algunos si chill o con chatgpt).

**-mv /home/usuario/dir1/f3 /home/usuario.** Tenes que tener permisos de root.

**- cp /home/usuario/dir2/f4 /home/usuario/dir1/dir11.** (se te copia el archivo a dentro de dir11)

**- cp /home/usuario/dir2/f4 /home/usuario/dir1/f7.** (el contenido de f4 se va a copiar en el archivo f7 (sigue estando el archivo f7 pero con el contenido de f4, si tenias algo en f7 te lo saca y copia lo que habia en f4).

**- cd /home/usuario.**

**mkdir copia**

**cp -r /home/usuario/dir1 /home/usuario/copia.**

**-no cambia nada los permisos.**

**- chmod 617 archivo.**

**- mv /home/usuario/dir1/f3 /home/usuario/dir1/f3.exe.**

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

`- chmod 023 /home/usuario/dir1/f3.exe ; chmod 023 /home/usuario/dir2/f4.exe.`

13) a) `mkdir /tmp/logs.`

b) `cp -r /var/log /tmp/logs.`

c) `tar -cf misLogs.tar -C /tmp logs.` **La opción -C se utiliza para cambiar el directorio actual antes de realizar la operación de empaquetado o extracción de manera temporal, ya que cuando ejecuto ese comando no estoy en /tmp, y no puedes poner /tmp/logs acá, vos deberías poner solo el nombre del archivo/directorio que quieres meterlo en misLogs. Lo que ocurre es que tar cambia al directorio /tmp, empaqueta el directorio "logs" en un archivo tar llamado "misLogs.tar" y luego vuelve al directorio original desde donde se ejecutó el comando. (SI SE PUEDE BLD)**

**Otra forma podría haber sido:** `cd/tmp; tar -cf misLogs.tar logs.`

d) `tar -czf misLogs.tar.gz -C /tmp logs.`

e) `cp /tmp/misLogs.tar /tmp/misLogs.tar.gz /home/usuario.`

f) `rm -r /tmp/logs.`

g) `cd ~.`

`mkdir direc1.`

`mkdir direc2.`

`tar -xf misLogs.tar -C direc1.`

`tar -xzf misLogs.tar.gz -C direc2.`

**En estos casos sin el -C no funciona, lo que hace ahí es que va de manera temporal a ese directorio y exporta todo lo que había en misLogs.tar o misLogs.tar.gz y vuelve al directorio actual.**

### **EXPLICANDO BIEN EL TAR:**

El comando ``tar`` se utiliza para crear y gestionar archivos de cinta (también llamados archivos tar). También es comúnmente utilizado para empaquetar y desempaquetar directorios y archivos en Linux.

La sintaxis básica del comando ``tar`` para crear un archivo tar es la siguiente:

```
``bash
tar -cf archivo_salida.tar archivos_o_directorios
``
```

Donde:

- ``-c``: Esta opción se utiliza para crear un archivo tar nuevo.

- ``-f archivo_salida.tar``: Esta opción especifica el nombre del archivo tar de salida que se creará.

- ``archivos_o_directorios``: Aquí puedes listar los archivos y/o directorios que deseas empaquetar en el archivo tar. Puedes especificar varios archivos o directorios separados por espacios.

Por ejemplo, si deseas empaquetar un directorio llamado "mi\_directorio" en un archivo tar llamado "mi\_archivo.tar", usarías el siguiente comando:

```
```bash
tar -cf mi_archivo.tar mi_directorio
```
```

Para empaquetar múltiples archivos y directorios en un solo archivo tar, simplemente los enumeras en la línea de comandos después de `-cf`. Por ejemplo:

```
```bash
tar -cf mi_archivo.tar archivo1 archivo2 directorio1
```
```

El comando anterior empaquetará "archivo1", "archivo2" y "directorio1" en un archivo tar llamado "mi\_archivo.tar".

Recuerda que el archivo tar creado contendrá una copia de los archivos y directorios, pero no los eliminará. Si deseas eliminar los archivos originales después de empaquetarlos, deberás hacerlo manualmente utilizando el comando `rm`.

El comando `tar` en Linux Debian admite varias combinaciones de opciones, además de `-cf`. Aquí te presento algunas de las combinaciones más comunes:

### 1. **\*\*Crear un archivo tar:\*\***

- `-c`: Crea un archivo tar nuevo.
- `-f archivo.tar`: Especifica el nombre del archivo tar de salida.

Ejemplo:

```
```bash
tar -cf archivo.tar directorio/
```
```

### 2. **\*\*Extraer un archivo tar:\*\***

- `-x`: Extrae los archivos de un archivo tar existente.
- `-f archivo.tar`: Especifica el archivo tar de entrada.

Te extrae todos los archivos que contiene, pero no te borra el archivo empaquetado, sigue estando. Y además, si te fijas en el contenido dentro del archivo empaquetado, los archivos/directorios que extrajiste siguen tmb estando ahí en el archivo empaquetado. (probado).

Ejemplo:

```
```bash
tar -xf archivo.tar
```
```

### 3. **\*\*Listar el contenido de un archivo tar:\*\***



- `-t`: Lista el contenido de un archivo tar sin extraerlo.
- `-f archivo.tar`: Especifica el archivo tar de entrada.

Ejemplo:

```
``bash
tar -tf archivo.tar
``
```

#### 4. **\*\*Agregar archivos o directorios a un archivo tar existente:\*\***

- `-r`: Agrega archivos o directorios a un archivo tar existente.
- `-f archivo.tar`: Especifica el archivo tar de entrada.

Ejemplo:

```
``bash
tar -rf archivo.tar nuevo_archivo
``
```

#### 5. **\*\*Comprimir un archivo tar:\*\***

- `-z`: Utilizado junto con `-c` para comprimir un archivo tar con gzip.
- `-j`: Utilizado junto con `-c` para comprimir un archivo tar con bzip2.

Ejemplos:

```
``bash
tar -czf archivo.tar.gz directorio/
tar -cjf archivo.tar.bz2 directorio/
``
```

**El comando tar puede invocar a gzip por nosotros (argumento "z"):**

**`tar -cvzf archivo.tar.gz arch1 arch2 arch3`**

**`tar -xvzf archivo.tar.gz`** El comando `tar -xvzf archivo.tar.gz` se utiliza para descomprimir y extraer el contenido de un archivo comprimido en formato tar.gz en Linux.

**-v:** La opción "v" significa "verbose" y permite que tar muestre información detallada sobre los archivos que se están incluyendo en el archivo comprimido. Esto puede ser útil para seguir el progreso de la creación del archivo y para verificar qué archivos se están incluyendo.

**Podes comprimir y descomprimir de manera suelta:**

- **Compresión:** Se reduce el tamaño de un archivo (gzip/bzip2/etc.)

```
gzip archivo.tar # Genera archivo.tar.gz comprimido
gzip -d archivo.tar.gz # Descomprime archivo.tar
```