Clases 4, 5 y 6

1. Buenas prácticas: comandos iniciales, conceptos y Guest Additions
2. El comando man
3. Atajos para moverse dentro de la terminal
4. Super usuario
5. Creación y gestión de usuarios y grupos
6. Comandos de directorio
7. Permisos en los archivos
8. Adición de discos a la máquina virtual
9. Comandos de configuraciones básicas
10. Tuberías
11. Redirigir la salida de un comando
12. Bash scripting
13. Interacción del hardware a nivel kernel
14. Gestión de Memoria de intercambio

1. Buenas prácticas: comandos iniciales, conceptos y Guest Additions

1.1 Es de las mejores prácticas como se ha descrito, antes de hacer cualquiera cosa, Clonar la PC.

1.2 Habilite las cantidades de recursos necesarios (Video, Micro, RAM, etc.)

1.3 Es siempre útil (dependiendo de los tiempos y recursos de la red) ejecutar **sudo apt update** (**En Fedora sería sudo dnf update / En CentOs sudo yum update**). Este comando actualiza la lista de paquetes disponibles en los repositorios de software configurados en tu sistema Ubuntu, asegurando que tengas la información más reciente sobre los paquetes disponibles.

sudo apt update && sudo apt upgrade

Acá nos detenemos para introducir los siguientes comandos y conceptos.

El operador && se utiliza en la línea de comandos de Unix y Linux para ejecutar el segundo comando solo si el primero se ejecuta correctamente, es decir, si no produce ningún error o fallo.

En el ejemplo, sudo apt update && sudo apt upgrade, el comando sudo apt upgrade solo se ejecutará si sudo apt update se ejecuta correctamente.

Para ejecutar los comandos de forma independiente, es decir, sin importar el resultado del otro, puedes separarlos usando un punto y coma; Por ejemplo:

sudo apt update; sudo apt upgrade

Con esto, ambos comandos se ejecutarán de manera independiente sin importar si uno de ellos falla o tiene éxito. Cada comando se ejecutará secuencialmente en el orden en que aparecen.

A continuación, se presentan los comandos más comunes para la gestión de paquetes en Linux:

apt update:

- Descripción: Este comando actualiza la lista de paquetes disponibles, descargando información sobre las últimas versiones de los paquetes desde los repositorios configurados en el sistema.

- Uso: `sudo apt update`

apt upgrade:

- Descripción: Actualiza todos los paquetes instalados en el sistema a la última versión disponible.

- Uso: `sudo apt upgrade`

apt autoremove:

- Descripción: Elimina automáticamente los paquetes que no son necesarios, es decir, aquellos que fueron instalados como dependencias de otros paquetes y que ya no son requeridos por ningún otro.

- Uso: `sudo apt autoremove`

apt install [paquete]:

- Descripción: Permite instalar un paquete específico en el sistema.

- Uso: `sudo apt install [nombre_paquete]`

apt remove [paquete]:

- Descripción: Desinstala un paquete del sistema, eliminando todos sus archivos y configuraciones asociadas.

- Uso: `sudo apt remove [nombre_paquete]`

1.4 También es una excelente práctica compartir bidireccionalmente para que se puede copiar y pegar textos, por ejemplo, entre el host el guest

1. **Selecciona tu máquina virtual** en la lista de máquinas virtuales en la ventana principal de VirtualBox.

2. **Haz clic en Configuración (Settings)** en la barra de menú superior o haz clic derecho en la máquina virtual y selecciona Configuración.
3. En la ventana de Configuración de la máquina virtual, **ve a la pestaña General**.
4. En la pestaña General, **ve a la sección Avanzado**.
5. Dentro de la sección Avanzado, **asegúrate de que la opción "Compartir Portapapeles"** esté configurada en "Bidireccional".

1.5 Las Guest Additions son un conjunto de controladores y herramientas que mejoran la experiencia de usuario dentro de una máquina virtual. Estos proporcionan funcionalidades adicionales, como la integración del ratón, la redimensionabilidad de la pantalla y el intercambio de portapapeles entre el sistema anfitrión y el invitado. Aquí te muestro cómo instalar las Guest Additions en una máquina virtual:

- Iniciar la máquina virtual: Inicia la máquina virtual en la que deseas instalar las Guest Additions.
- Montar la imagen de Guest Additions: En el menú de VirtualBox, haz clic en "Dispositivos" y luego en "Insertar imagen de CD de Guest Additions". Esto montará la imagen de las Guest Additions en la máquina virtual como un CD virtual.
- Acceder al CD virtual: Accede al CD virtual dentro de la máquina virtual. Dependiendo del sistema operativo invitado, es posible que necesites montar el CD manualmente.
- Ejecutar el instalador: Una vez que hayas accedido al CD virtual, busca el archivo de instalación de las Guest Additions (por lo general, se llama "VBoxWindowsAdditions.exe" para Windows o "VBoxLinuxAdditions.run" para Linux) y ejecútalo.
- Seguir el proceso de instalación: Sigue las instrucciones del instalador para completar el proceso de instalación de las Guest Additions. Es posible que necesites reiniciar la máquina virtual después de la instalación. Y, es posible que necesites instalar sudo apt install bzip2 (Seguimos introduciendo comandos acá mandamos a instar con privilegios un compactador de archivos)
- Comprobar la instalación: Una vez que se complete la instalación, puedes comprobar si las Guest Additions están instaladas y funcionando correctamente. Por ejemplo, en una máquina virtual de Windows, verás la resolución de la pantalla ajustarse automáticamente al cambiar el tamaño de la ventana de VirtualBox.

Para verificar si tienes las "Guest Additions" instaladas en tu máquina virtual por comando

1. Inicia tu máquina virtual Ubuntu.

2. Una vez que Ubuntu haya arrancado, inicia sesión en tu cuenta.

3. Abre una terminal. Puedes hacerlo buscando "Terminal" en el menú de aplicaciones o presionando `Ctrl + Alt + T`. (Detalle de teclas rápidas)

4. En la terminal, ejecuta el siguiente comando para verificar si las Guest Additions están instaladas:

lsmod | grep vboxguest

Este comando buscará el módulo del kernel `vboxguest`, que es parte de las Guest Additions de VirtualBox. Si el comando devuelve alguna salida, significa que las Guest Additions están instaladas y cargadas en el kernel.

Aquí introducimos dos comandos más y el concepto de tubería… Dejamos una explicación inicial:

Aquí está el significado de cada parte del comando:

- **lsmod**: Muestra una lista de todos los módulos del kernel actualmente cargados en el sistema.
- **|**: El símbolo de tubería (**|**) se utiliza para redirigir la salida del comando **lsmod** al siguiente comando en la secuencia.
- **grep**: Es un comando utilizado para buscar patrones en la entrada proporcionada.
- **vboxguest**: Es el patrón que estamos buscando. En este caso, queremos encontrar todas las líneas que contienen "vboxguest".

La tubería, representada por el símbolo "|", es un concepto fundamental en los sistemas operativos tipo Unix y Linux que permite enviar la salida de un comando como entrada a otro comando. Esto significa que puedes conectar la salida de un comando con la entrada de otro, creando así una secuencia de procesos en la que la salida de uno se convierte en la entrada del siguiente.

2- El comando man

- **¿Qué son los manuales de comandos?**
  - Los manuales de comandos son documentos que proporcionan información detallada sobre cómo usar y comprender los comandos en un sistema operativo Unix/Linux.
- **Comando man: Mostrando el manual del comando**
  - **man**: El comando **man** se utiliza para mostrar el manual de un comando específico.
  - Ejemplo: **man man** muestra el manual del comando **man** mismo.
  - Ejemplo: **man passwd** muestra el manual del comando **passwd**.
- **Comando apropos: Buscando comandos por palabra clave**
  - **apropos**: Este comando se utiliza para buscar comandos relacionados con una palabra clave.

- Ejemplo: **apropos network** busca comandos relacionados con la palabra clave "network".

Explorando las secciones del manual

- **Sección 1: Programas ejecutables u órdenes de la shell**
  - Ejemplo: **man Shell** muestra el manual del comando **man**.
- **Sección 2: Llamadas al sistema (funciones proporcionadas por el núcleo)**
  - Ejemplo: **man open** muestra el manual de la llamada al sistema **open**.
- **Sección 3: Llamadas a biblioteca (funciones dentro de bibliotecas de programa)**
  - Ejemplo: **man stdio** muestra el manual de las funciones de la biblioteca estándar de E/S.
- **Sección 4: Archivos especiales (normalmente se encuentran en /dev)**
  - Ejemplo: **man tty** muestra el manual del archivo especial **tty**.
- **Sección 5: Formatos de archivo y convenios**
  - Ejemplo: **man 5 passwd** muestra el manual del formato de archivo **/etc/passwd**.
- **Sección 6: Juegos**
  - Ejemplo: **man 6 fortune** muestra el manual del juego de palabras **fortune**.
- **Sección 7: Miscelánea (incluidos paquetes de macros y convenios)**
  - Ejemplo: **man ascii** muestra el manual del estándar ASCII.
- **Sección 8: Órdenes de administración del sistema (normalmente solo para root)**
  - Ejemplo: **man sudo** muestra el manual del comando **sudo**.
- **Sección 9: Rutinas del núcleo [No estándar]**
  - Ejemplo: **man kmalloc** muestra el manual de la rutina del núcleo **kmalloc**.

Búsqueda y filtrado de comandos

- **Buscando comandos relacionados**
  - Ejemplo: **man -k network** busca todos los comandos relacionados con la red.
  - Ejemplo: **apropos "network devices" | grep ip** busca comandos relacionados con dispositivos de red que contienen "ip".

3 - Atajos para Moverse Dentro de la Terminal:

- Ctrl + E: Moverse al final de la línea.
- Ctrl + A: Moverse al principio de la línea.
- Ctrl + → y Ctrl + ←: Saltar de palabra en palabra hacia adelante y hacia atrás, respectivamente.

- Ctrl + W: Eliminar la palabra anterior.
- Ctrl + U: Eliminar desde la posición actual hasta el principio de la línea.
- Ctrl + K: Eliminar desde la posición actual hasta el final de la línea.
- Ctrl + T: Intercambiar las dos letras antes del cursor.
- Alt + T: Intercambiar las dos palabras antes del cursor.
- Ctrl + Y: Pegar texto previamente eliminado.

Atajos para Búsqueda en el Historial:

- history: Mostrar el historial completo de comandos.
- Ctrl + R: Realizar una búsqueda inversa en el historial de comandos.
- !número: Ejecutar el comando con el número específico en el historial.
- !!: Ejecutar el último comando.
- sudo !!: Ejecutar el último comando con privilegios de superusuario.
- Facilita la búsqueda de comandos anteriores en la historia de la terminal y agiliza la ejecución con estos atajos específicos.

4 – Super usuario

- **Introducción a los privilegios de superusuario**
  - En el mundo de Linux, el superusuario, también conocido como **root**, es el usuario con los máximos privilegios en el sistema.
  - El superusuario tiene acceso completo a todos los archivos y configuraciones del sistema, lo que le permite realizar cambios críticos y administrativos.
- **Comandos útiles**
  - **whoami**: Este comando muestra el nombre del usuario actual, lo que es útil para verificar si se está utilizando el superusuario.
  - **id**: Proporciona información detallada sobre el usuario y su grupo, incluidos los privilegios del superusuario.
  - **w**: Muestra información sobre los usuarios conectados al sistema y sus actividades actuales.
  - **last**: Ofrece un historial detallado de los inicios de sesión anteriores en el sistema.
- **Utilizando sudo para obtener privilegios de superusuario**
  - **sudo**: El comando **sudo** permite a los usuarios ejecutar comandos con los privilegios del superusuario temporalmente.
  - **sudo su**: Con **sudo su**, un usuario puede cambiar al superusuario y ejecutar múltiples comandos con esos privilegios sin tener que ingresar la contraseña de nuevo.

5. Creación y gestión de usuarios y grupos

- **useradd**: Este comando se utiliza para agregar un nuevo usuario al sistema.

- **groupadd**: Permite crear un nuevo grupo de usuarios en el sistema.
- **usermod**: Proporciona opciones para modificar las propiedades de un usuario existente, como su nombre, directorio principal, etc.

Practicando:
Creo el usuario yo
Sudo useradd yo

Creo el grupo prueba
Sudo groupadd prueba

Añado el usuario yo al grupo prueba sin afectarlo
sudo usermod -aG prueba yo

Muestro la información sobre el usuario yo
Id yo

Pongo password al usuario yo
Sudo passwd yo

6 - Comandos de directorio

Los sistemas operativos basados en Linux ofrecen una amplia gama de comandos para la gestión de directorios y archivos. Estos comandos son fundamentales para la navegación, creación, manipulación y búsqueda de archivos en el sistema de archivos.

pwd (Print Working Directory):

- Descripción: Este comando muestra la ruta completa del directorio actual en el que te encuentras.

- Uso: `pwd`

ls (List Directory Contents):

Descripción: Lista el contenido del directorio actual.

Opciones principales:

-a: Muestra todos los archivos, incluidos los ocultos.

-l: Muestra detalles largos, incluyendo permisos, propietario, grupo y tamaño.

-h: Muestra tamaños de archivo legibles para humanos (en kilobytes, megabytes, etc.).

cd (Change Directory):

- Descripción: Cambia el directorio actual al especificado.

- Uso: `cd [directorio]`

tree:

Descripción: Muestra la estructura de directorios en forma de árbol.

Opciones principales:

-L [n]: Limita la profundidad del árbol a n niveles.

-d: Solo muestra directorios.

mkdir (Make Directory):

- Descripción: Crea un nuevo directorio.

- Uso: `mkdir [nombre_directorio]`

rm (Remove):

- Descripción: Elimina archivos o directorios.

- Uso: `rm [archivo/directorio]`

touch:

- Descripción: Crea archivos vacíos o actualiza las marcas de tiempo de los archivos existentes.

- Uso: `touch [nombre_archivo]`

cp (Copy)

- Descripción: Copia archivos o directorios.

- Opciones principales:

  - `-r` o `--recursive`: Copia directorios de forma recursiva.

  - `-v` o `--verbose`: Muestra información detallada.

- Ejemplo de copia recursiva del directorio:

  cp -r directorio_origen directorio_destino

mv (Move):

- Descripción: Mueve o renombra archivos o directorios.

- Ejemplo de mover un archivo:

  mv archivo.txt /ruta/nuevo_directorio/

- Ejemplo de renombrar un archivo:

  mv archivo_antiguo.txt archivo_nuevo.txt

Búsqueda y Localización de Archivos:

locate:

- Descripción: Busca archivos o directorios en la base de datos del sistema.

- Ejemplo:

  locate archivo.txt

updatedb:

- Descripción: Actualiza la base de datos de `locate` para reflejar los cambios en el sistema de archivos.

whereis:

- Descripción: Muestra ubicaciones de archivos binarios, fuentes y páginas de manuales.

- Ejemplo:

  whereis nano

whatis:

- Descripción: Muestra una descripción breve de un comando.

- Ejemplo:

  whatis ls

El comando whereis nano debería devolver las ubicaciones de los archivos binarios, fuentes y páginas de manual asociadas con el editor de texto nano.

Por otro lado, el comando whatis ls debería devolver una descripción breve del comando ls, que es utilizado para listar el contenido de un directorio en sistemas Unix y similares.

find:

- Descripción: Busca archivos y directorios en función de varios criterios.

- Opciones principales:

  - `-name [nombre_archivo]`: Especifica el nombre del archivo a buscar.

- Ejemplo:

  find /home/usuario -name "*.txt"

dmesg:

- Descripción: Muestra el buffer de mensajes del kernel, útil para diagnosticar problemas durante el arranque del sistema.

- Ejemplo:

  dmesg **| grep error**

fdisk:

- Descripción: Herramienta para manipular la tabla de particiones en sistemas Unix y Linux.

- Ejemplo:

  fdisk /dev/sda

mkfs.ext4:

- Descripción: Crea un sistema de archivos ext4 en una partición específica.

- Ejemplo:

  mkfs.ext4 /dev/sda1

df (Disk Free):

- Descripción: Muestra el espacio utilizado y disponible en los sistemas de archivos montados.

- Opciones principales:

  - `-h`: Muestra tamaños de archivo legibles para humanos (en kilobytes, megabytes, etc.).

- Ejemplo:

  df -h

lsblk (List Block Devices):

- Descripción: Lista información sobre los dispositivos de bloque, incluidas sus particiones.

- Opciones principales: No tiene opciones adicionales.

- Ejemplo:

  lsblk

mount:

- Descripción: Monta un sistema de archivos en una ubicación específica del árbol de directorios.

- Opciones principales: No tiene opciones adicionales.

- Ejemplo:

mount /dev/sdb1 /mnt/usb

umount:

- Descripción: Desmonta un sistema de archivos previamente montado.

- Opciones principales: No tiene opciones adicionales.

- Ejemplo:

umount **/mnt/usb**

7 - Permisos en los archivos

En el sistema operativo Linux, los permisos de archivos y directorios son fundamentales para controlar quién puede acceder, leer, escribir o ejecutar esos archivos. Estos permisos están diseñados para garantizar la seguridad y la privacidad de los datos almacenados en el sistema.

Los permisos se dividen en tres categorías principales:

Usuario propietario: Se refiere al propietario del archivo o directorio. Este usuario tiene control total sobre el archivo y puede cambiar sus permisos, propietario y grupo.
Grupo: Se refiere al grupo al que pertenece el archivo o directorio. Todos los usuarios que pertenezcan a este grupo tendrán los mismos permisos sobre el archivo o directorio, aunque no sean el propietario.
Otros usuarios: Se refiere a todos los usuarios que no son el propietario ni pertenecen al grupo del archivo o directorio. Estos usuarios tienen los permisos más limitados y solo pueden realizar ciertas acciones según los permisos establecidos.

Comandos útiles para administrar permisos de archivos

- **chmod**: Este comando se utiliza para cambiar los permisos de archivos y directorios en Linux. Puede cambiar los permisos agregando o quitando permisos para el usuario propietario, el grupo y otros usuarios.
- **chown**: Permite cambiar el propietario y el grupo de un archivo o directorio. Esto es útil cuando se desea transferir la propiedad de un archivo a otro usuario o grupo.
- **chgrp**: Se utiliza para cambiar el grupo de un archivo o directorio. Esto puede ser útil para asignar archivos a un grupo específico y controlar el acceso de los usuarios que pertenecen a ese grupo.

Así:

chmod: Cambia los permisos de lectura, escritura y ejecución de archivos y directorios.

Ejemplo: Dar permisos de lectura, escritura y ejecución al propietario del archivo

chmod u+rwx archivo.txt

chgrp: Cambia el grupo de un archivo o directorio.

Ejemplo: Cambiar el grupo de un archivo

chgrp grupo_nuevo archivo.txt

chown: Cambia el propietario y el grupo de un archivo o directorio.

Ejemplo: Cambiar el propietario y el grupo de un archivo

chown nuevo_propietario:nuevo_grupo archivo.txt

Interpretación de permisos en la salida de ls

Cuando se utiliza el comando **ls -l** para listar archivos y directorios, la salida muestra una serie de caracteres que representan los permisos de archivo. Por ejemplo: **-rwxr-xr--**.

- El primer carácter indica el tipo de archivo (por ejemplo, **-** para un archivo regular, **d** para un directorio, **l** para un enlace simbólico, etc.).
- Los siguientes nueve caracteres se dividen en tres conjuntos de tres caracteres cada uno. Cada conjunto representa los permisos para el propietario, el grupo y otros usuarios, respectivamente.

Por ejemplo, **rwxr-xr--** significa que el propietario tiene permisos de lectura, escritura y ejecución, el grupo tiene permisos de lectura y ejecución, y otros usuarios solo tienen permisos de lectura.

Practicando:

Sino se creó la carpeta /home/yo la creo
Sudo mkdir /home/yo

Esto cambiará el propietario y el grupo de la carpeta

sudo chown yo:prueba /home/yo

Voy a crear ahora la carpeta /home/guille
Sudo mkdir /home/guille

Y voy a dar permisos al grupo prueba sobre la carpeta /home/guille
sudo chgrp -R prueba /home/guille
sudo chmod -R g+rwx /home/guille

También pude dar permiso al usuario yo sin cambiar el grupo
sudo chmod -R u+rwx /home/guille

Y con esto mostramos que las opciones de chmod son:

u: Hace referencia al usuario propietario del archivo o directorio.
g: Hace referencia al grupo propietario del archivo o directorio.
o: Hace referencia a los demás usuarios (otros) que no son el propietario ni están en el grupo propietario del archivo o directorio.
a: Hace referencia a todos los usuarios, incluyendo al propietario, al grupo y a los demás usuarios.
Estas opciones se combinan con los símbolos de permisos:

+: Agrega los permisos especificados.
-: Quita los permisos especificados.
=: Establece exactamente los permisos especificados, sobrescribiendo los permisos existentes.

Por ejemplo:
u+rwx: Agrega permisos de lectura, escritura y ejecución al usuario propietario.
g-rw: Quita los permisos de lectura y escritura al grupo propietario.
o=r: Establece permisos de lectura para otros usuarios, sobrescribiendo los permisos existentes.
a+x: Agrega permisos de ejecución para todos los usuarios.

En Linux, además de los permisos básicos de lectura, escritura y ejecución, existen permisos especiales que permiten configuraciones más avanzadas de seguridad y acceso. Estos permisos especiales son:

1. **SUID (Set User ID)**:

   - Cuando el SUID está establecido en un archivo ejecutable, el programa se ejecuta con los mismos privilegios que el propietario del archivo, independientemente de quién lo ejecute. Por lo general, se utiliza en casos

donde un programa necesita acceso a recursos que solo el propietario tiene, como cambiar contraseñas o realizar tareas de administración del sistema.

- Para establecer el SUID en un archivo, se usa el bit de permisos "s" en lugar de "x" para el usuario propietario. Por ejemplo, **chmod u+s archivo**.

2. **SGID (Set Group ID)**:

- Cuando el SGID está establecido en un archivo ejecutable, el programa se ejecuta con los mismos privilegios que el grupo propietario del archivo, independientemente de quién lo ejecute. Esto es útil para aplicaciones que necesitan acceso a archivos o recursos que solo están disponibles para un grupo específico.

- Para establecer el SGID en un archivo, se usa el bit de permisos "s" en lugar de "x" para el grupo propietario. Por ejemplo, **chmod g+s archivo**.

3. **Sticky Bit**:

- Cuando el Sticky Bit está establecido en un directorio, solo el propietario del archivo, el propietario del directorio y el usuario root pueden eliminar o renombrar los archivos dentro de ese directorio, incluso si otros usuarios tienen permisos de escritura en el directorio.

- Para establecer el Sticky Bit en un directorio, se usa el bit de permisos "t". Por ejemplo, **chmod +t directorio**.

Otros archivos de configuración relevantes son

- **/etc/passwd**: Contiene información sobre todas las cuentas de usuario registradas en el sistema, incluidos los nombres de usuario y los identificadores de usuario (UID).
- **/etc/shadow**: Almacena las contraseñas cifradas de los usuarios del sistema, garantizando su seguridad y confidencialidad.
- **/etc/group**: Contiene detalles sobre los grupos de usuarios en el sistema, como sus nombres y los identificadores de grupo (GID).

Podemos mostrar los usuarios del sistema con el comando cat para que imprima el contenido del archivo:
cat /etc/passwd

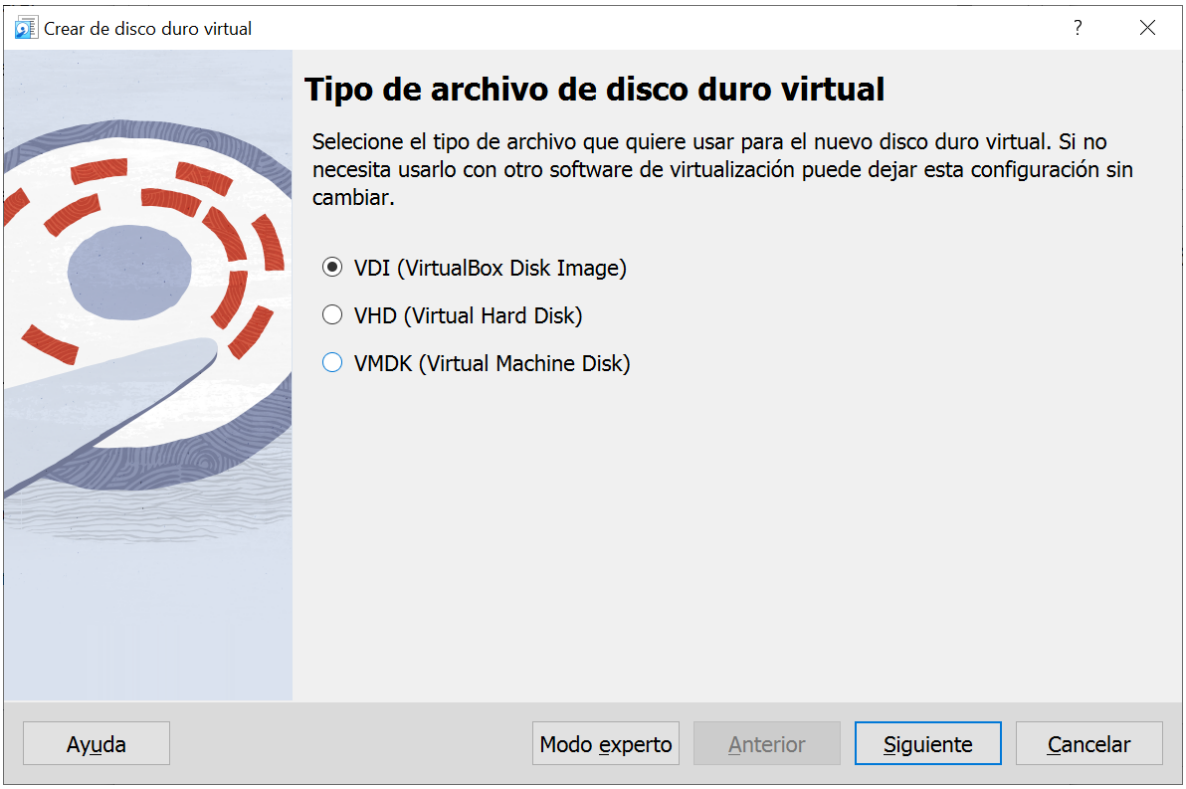
Podemos usar el comando cut para filtrar solo el nombre de usuario de la salida del archivo /etc/passwd:

cat /etc/passwd | cut -d ":" -f 1

8- Adición de discos a la máquina virtual

La adición de discos a una máquina virtual en VirtualBox es un proceso sencillo que te permite aumentar la capacidad de almacenamiento de la máquina virtual. Aquí están los pasos para hacerlo:

- Abrir VirtualBox: Inicia VirtualBox en tu sistema operativo.
- Seleccionar la máquina virtual: Selecciona la máquina virtual a la que deseas añadir un disco en la lista de máquinas virtuales.
- Acceder a configuración: Haz clic en el botón "Configuración" en la parte superior de la ventana de VirtualBox para abrir la configuración de la máquina virtual seleccionada.
- Agregar disco: En la ventana de configuración de la máquina virtual, selecciona la pestaña "Almacenamiento". Allí podrás ver los dispositivos de almacenamiento actuales de la máquina virtual.
- Agregar controlador de disco: Si deseas añadir un nuevo disco duro, haz clic en el icono de "+" junto al controlador SATA o IDE para añadir un nuevo disco.
- Configurar disco: Se abrirá un asistente para añadir un nuevo disco. Aquí puedes crear un nuevo disco virtual o seleccionar un disco existente en tu sistema. Sigue las instrucciones del asistente para configurar el disco según tus necesidades.
- Finalizar: Una vez que hayas configurado el disco, haz clic en "Aceptar" o "OK" para cerrar la ventana de configuración de la máquina virtual.



Normalmente el disco no se monta en el sistema operativo por lo que ejecutamos el comando (Seguimos introduciendo comandos) lsblk

Vemos como se llama, supón sdb

Creamos donde lo vamos a montar
sudo mkdir /mnt/disco

Lo particionamos
sudo mkfs.ext4 /dev/sdb

Lo montamos
sudo mount /dev/sdb /mnt/disco

9- Comandos de configuraciones básicas

En el entorno de Linux, la configuración del sistema y de los usuarios es una parte fundamental para garantizar un funcionamiento óptimo y seguro. Aquí se presentan una serie de comandos y archivos de configuración básicos que son esenciales para personalizar y administrar un sistema Linux:

hostnamectl set-hostname pc-pepe:

- Descripción: Este comando se utiliza para cambiar el nombre de la PC.

- Uso: `hostnamectl set-hostname [nuevo_nombre]`

localectl set-keymap us:

- Descripción: Cambia la configuración del teclado a inglés.

- Uso: `localectl set-keymap us`

localectl set-keymap latam:

- Descripción: Cambia la configuración del teclado a español latino.

- Uso: `localectl set-keymap latam`

/etc/passwd:

- Descripción: Archivo que almacena información de cuentas de usuario, como nombres de usuario, ID de usuario, ID de grupo, directorio principal y shell.

/etc/shadow:

- Descripción: Archivo que almacena las contraseñas encriptadas de los usuarios.

/etc/hosts:

- Descripción: Archivo que mapea direcciones IP a nombres de host. Se utiliza para resolver nombres de host localmente.

/etc/sudoers:

- Descripción: Archivo que especifica qué usuarios tienen permiso para ejecutar comandos como superusuario mediante sudo.

Configuración del Usuario:

$HOME/.ssh/config:

- Descripción: Archivo de configuración personalizada para conexiones SSH.

$HOME/.bashrc:

- Descripción: Archivo de configuración de inicio de la shell Bash, se ejecuta cada vez que se inicia una nueva instancia de la shell.

$HOME/.bash_history:

- Descripción: Historial de comandos ejecutados en la shell Bash, útil para recordar comandos anteriores.

Logs del Sistema:

/var/log/messages:

- Descripción: Registro de mensajes del sistema, que incluye información sobre eventos del sistema, errores, advertencias y otras actividades.

`uname -a`:

- Descripción: Muestra información sobre el kernel y el sistema operativo, como el nombre del kernel, la versión y la arquitectura del sistema.

`cat /proc/cpuinfo`:

- Descripción: Muestra información detallada sobre la CPU del sistema, incluyendo el modelo, velocidad y número de núcleos.

`free -m`:

- Descripción: Muestra la cantidad de memoria RAM disponible y en uso en el sistema.

`df -h`:

- Descripción: Muestra el espacio disponible en los sistemas de archivos montados de forma legible para el usuario.

`lsb_release -a`:

- Descripción: Muestra información sobre la distribución Linux y su versión.

`lspci`:

- Descripción: Muestra información sobre los dispositivos PCI conectados al sistema.

`lsblk`:

- Descripción: Lista información sobre los dispositivos de bloque, incluyendo sus particiones.

`ifconfig`:

- Descripción: Muestra la configuración de red de las interfaces de red en el sistema.

`lscpu`:

- Descripción: Muestra información detallada sobre la CPU del sistema, incluyendo la arquitectura y las características del procesador.

`lsusb`:

- Descripción: Lista los dispositivos USB conectados al sistema.

`uptime`:

- Descripción: Muestra el tiempo que ha estado funcionando el sistema desde su última puesta en marcha.

`ps aux`:

- Descripción: Muestra una lista detallada de todos los procesos en ejecución en el sistema.

Kill

- Descripción: Detener un proceso utilizando su ID de proceso (PID):

kill PID

Por otro lado, **killall** es un comando similar a **kill**, pero en lugar de requerir un PID, utiliza el nombre del proceso para detenerlo. Esto puede ser útil cuando se desea detener varios procesos con el mismo nombre.

Ejemplo de uso de **killall**:

`top`:

- Descripción: Proporciona una visualización en tiempo real de los procesos en ejecución y la carga del sistema.

`cat`, `grep`, `less`, `tail`, `head`, `more`:

- Descripción: Estos comandos se utilizan para explorar archivos de configuración y buscar información en ellos.

Por supuesto, aquí tienes una descripción de cada uno de los comandos junto con ejemplos de su uso:

cat:

- Descripción: El comando `cat` se utiliza para concatenar y mostrar el contenido de uno o más archivos en la salida estándar. Es útil para visualizar el contenido completo de un archivo en la terminal.

- Ejemplo: cat archivo.txt

grep:

- Descripción: `grep` se utiliza para buscar patrones de texto dentro de uno o varios archivos. Puede buscar líneas que coincidan con un patrón específico y mostrarlas en la salida estándar.

- Ejemplo: grep "palabra" archivo.txt

less:

- Descripción: `less` es un visor de texto que permite desplazarse hacia arriba y hacia abajo a través de archivos grandes de forma más eficiente que `cat`. Proporciona una visualización más amigable y permite buscar dentro del archivo.

- Ejemplo: less archivo_grande.log

tail:

- Descripción: `tail` muestra las últimas líneas de un archivo de texto. Es útil para monitorear archivos de registro en tiempo real.

- Ejemplo: tail archivo.log

head:

- Descripción: `head` muestra las primeras líneas de un archivo de texto. Es útil para ver un resumen inicial de un archivo.

- Ejemplo: head archivo.txt

more:

- Descripción: `more` es similar a `less`, pero muestra el contenido del archivo de forma paginada, permitiendo al usuario navegar por él de forma más controlada. Es menos flexible que `less` pero está disponible en sistemas Unix más antiguos.

- Ejemplo: more archivo_grande.log

10- Tuberías qué son y ejemplos

En Linux, las tuberías son una característica fundamental que permite la comunicación entre procesos, facilitando el flujo de datos de la salida estándar (stdout) de un proceso a la entrada estándar (stdin) de otro proceso. Esto se logra mediante el uso del operador de tubería `|`, que conecta la salida de un comando con la entrada de otro.

Concepto:

- Las tuberías permiten la transferencia de datos entre procesos de manera eficiente, sin necesidad de guardar temporalmente la salida en archivos intermedios.

- Se utilizan para encadenar múltiples comandos y realizar operaciones complejas en una sola línea de comando.

Ejemplos:

Contar las líneas de un archivo:

```
cat archivo.txt | wc -l
```

- `cat archivo.txt` muestra el contenido del archivo.
- `wc -l` cuenta las líneas de entrada.

Buscar un archivo y mostrar solo el nombre del archivo:

```
find /ruta -name "archivo*" | xargs basename
```

- `find /ruta -name "archivo*"` busca archivos que coincidan con el patrón en la ruta especificada.
- `xargs basename` muestra solo el nombre de archivo de la salida de `find`.

Ordenar y mostrar el contenido único de un archivo:

```
sort archivo.txt | uniq
```

- `sort archivo.txt` ordena las líneas del archivo.
- `uniq` muestra solo las líneas únicas consecutivas.

Filtrar la salida de un comando:

```
ps aux | grep "proceso"
```

- `ps aux` muestra todos los procesos en ejecución.
- `grep "proceso"` filtra las líneas que contienen la palabra "proceso".

Contar la cantidad de archivos en un directorio:

```
ls -l | grep "^-" | wc -l
```

- `ls -l` lista el contenido del directorio en formato detallado.
- `grep "^-"` filtra las líneas que comienzan con "-", que indica archivos (no directorios).
- `wc -l` cuenta las líneas filtradas, es decir, la cantidad de archivos.

Los comandos utilizados en las tuberías pueden incluir cualquier comando de la línea de comandos de Linux, incluidos aquellos relacionados con la gestión de paquetes, configuraciones básicas, administración de archivos y directorios, entre otros.

- Es importante comprender los permisos de ejecución de los comandos involucrados y utilizar `sudo` cuando sea necesario para obtener privilegios de superusuario.

- Las tuberías ofrecen una forma poderosa y versátil de realizar tareas complejas mediante la combinación de múltiples comandos en una única secuencia de ejecución.

11- Redirigir la salida de un comando

En Linux, es común redirigir la salida de un comando a un archivo en lugar de mostrarla en la pantalla. Esto es útil para guardar resultados, registrar información o procesar datos más adelante.

1. Redirigir la salida estándar a un archivo

Puedes redirigir la salida estándar (stdout) de un comando a un archivo usando el operador >.

Ejemplo:

ls > listado_directorio.txt

Explicación:

- **ls**: Muestra el contenido del directorio actual.
- **>**: Redirige la salida del comando a un archivo.
- **listado_directorio.txt**: Es el archivo en el que se guardará la salida del comando ls.

Si el archivo listado_directorio.txt no existe, se creará. Si ya existe, se sobrescribirá con la nueva salida.

2. Redirigir la salida estándar y añadir al final del archivo

Para añadir la salida al final de un archivo sin sobrescribir su contenido, usa el operador >>.

Ejemplo:

df >> espacio_disco.txt

Explicación:

- **df**: Muestra información sobre el uso del sistema de archivos.
- **>>**: Redirige la salida y añade la información al final del archivo especificado.
- **espacio_disco.txt**: El archivo en el que se añade la salida del comando df.

3. Usar tee para redirigir la salida a un archivo y a la pantalla

El comando tee permite redirigir la salida a un archivo y al mismo tiempo mostrarla en la pantalla.

Ejemplo:

ls | tee listado_directorio.txt

Explicación:

- **ls**: Muestra el contenido del directorio.
- **|**: Pasa la salida del comando ls al siguiente comando. (El concepto de tubería)
- **tee**: Lee de la entrada estándar y escribe en la salida estándar y en el archivo.

12. Bash scripting

Bash Scripting es la práctica de escribir programas o secuencias de comandos utilizando el shell Bash (Bourne Again SHell). Bash es una interfaz de línea de comandos y un lenguaje de programación que permite a los usuarios interactuar con el sistema operativo y ejecutar tareas automatizadas.

Características principales:

1. Automatización de tareas: Bash scripting permite automatizar procesos repetitivos, como copiar archivos, ejecutar comandos en secuencia, administrar sistemas, entre otros.

2. Interacción con el sistema: Puedes interactuar directamente con el kernel del sistema operativo, realizar operaciones en archivos, manipular texto, y ejecutar comandos del sistema.

3. Control de flujo: Incluye estructuras de control como bucles (`for`, `while`), condicionales (`if`, `case`), y funciones para organizar y reutilizar código.

Estructura básica de un script Bash

1. Shebang (`#!/bin/bash`): Indica al sistema que el archivo debe ejecutarse usando el intérprete Bash.

2. Comandos del sistema: Puedes incluir cualquier comando de terminal que usarías manualmente.

3. Variables y operadores: Se pueden definir variables para almacenar información y operadores para realizar operaciones matemáticas o lógicas.

Estructura if

La estructura if permite ejecutar comandos si se cumple una condición. Es fundamental para controlar el flujo de cualquier script.

Forma 1: Ejecutar directamente en la terminal

```
echo "Introduce un número:"
read numero
if [ $numero -gt 10 ]; then
    echo "El número es mayor que 10."
else
    echo "El número es menor o igual que 10."
fi
```

Forma 2: Creando un archivo Bash

Crea un archivo llamado ejemplo.sh:

nano if_example.sh

Escribe este código dentro del archivo:

```
#!/bin/bash

echo "Introduce un número:"

read numero

if [ $numero -gt 10 ]; then

    echo "El número es mayor que 10."

else

    echo "El número es menor o igual que 10."

fi
```

Ejecuta el script:

```
./ejemplo.sh
```

2. Estructura for

El bucle for es útil cuando necesitas iterar sobre una lista de elementos. Veamos cómo funciona.

```
for i in 1 2 3 4 5; do

    echo "Número: $i"

done
```

3. Estructura while

El bucle while se repite mientras se cumpla una condición. Es útil para ejecutar acciones hasta que la condición cambie.

```
contador=1
while [ $contador -le 5 ]; do
    echo "Contador: $contador"
    contador=$((contador + 1))
done
```

4. Estructura case

case es una estructura que permite comparar un valor con diferentes patrones y ejecutar comandos dependiendo de la coincidencia.

```
echo "Introduce una opción (start/stop/restart):"
read opcion
case $opcion in
    start)
        echo "Iniciando el servicio..."
        ;;
    stop)
        echo "Deteniendo el servicio..."
        ;;
    restart)
        echo "Reiniciando el servicio..."
        ;;
    *)
        echo "Opción no válida."
```

;;

esac

13. Interacción del hardware a nivel kernel

Para mostrar la interacción del hardware a nivel kernel en Linux, vamos utilizar varias herramientas y archivos del sistema que permiten observar cómo el kernel maneja y se comunica con los dispositivos de hardware.

1. Archivos del sistema /proc y /sys

- **/proc**: Contiene información sobre procesos y hardware gestionados por el kernel.
- **/sys**: Expone detalles sobre los dispositivos y módulos del kernel.

Ejemplo 1: Información de la CPU

Para ver la información sobre la CPU que el kernel está gestionando, puedes inspeccionar el archivo /proc/cpuinfo.

cat /proc/cpuinfo

Esto mostrará detalles como el modelo de la CPU, el número de núcleos, la velocidad del procesador, entre otros datos que el kernel utiliza para interactuar con el procesador.

Ejemplo 2: Información de la memoria

El archivo /proc/meminfo contiene información sobre el uso de la memoria en el sistema, que el kernel está monitoreando.

cat /proc/meminfo

Esto mostrará detalles sobre la memoria total, la memoria libre, la memoria en caché, entre otros parámetros gestionados por el kernel.

Ejemplo 3: Dispositivos conectados al sistema

El directorio /sys contiene información sobre los dispositivos físicos conectados al sistema y cómo el kernel interactúa con ellos. Por ejemplo, para ver los dispositivos USB:

ls /sys/bus/usb/devices/

Para más detalles de un dispositivo USB específico:

cat /sys/bus/usb/devices/usb1/product

Esto te mostrará el nombre del dispositivo conectado.

2. Dmesg

El comando dmesg imprime los mensajes del kernel, incluidos aquellos relacionados con la interacción del hardware. Cada vez que se conecta o desconecta un dispositivo, o cuando hay un problema relacionado con hardware, el kernel genera un mensaje.

Ejemplo 4: Ver mensajes del kernel relacionados con hardware

dmesg | grep -i "usb"

Este comando filtrará todos los mensajes relacionados con dispositivos USB. Puedes cambiar "usb" por otros términos según el hardware que quieras observar (como "ethernet", "pci", etc.).

Ejemplo 5: Mensajes relacionados con el disco duro

dmesg | grep -i "sda"

Esto mostrará mensajes relacionados con el disco duro /dev/sda, como detección de particiones, errores de lectura/escritura, etc.

3. Lspci y lsusb

Los comandos lspci y lsusb son útiles para listar dispositivos PCI y USB, respectivamente.

Ejemplo 6: Listar dispositivos PCI

lspci

Este comando te mostrará una lista de todos los dispositivos conectados al bus PCI, como tarjetas de red, tarjetas gráficas, controladores SATA, etc.

Ejemplo 7: Listar dispositivos USB

lsusb

Este comando te mostrará una lista de todos los dispositivos conectados al bus USB.

4. Interacción con el Hardware mediante Módulos del Kernel

Los módulos del kernel permiten gestionar de manera modular los controladores de hardware. Puedes cargar y descargar módulos para interactuar con el hardware.

Ejemplo 8: Listar módulos del kernel cargados

lsmod

Esto muestra todos los módulos del kernel que están actualmente cargados y utilizados por el sistema para interactuar con el hardware.

Ejemplo 9: Cargar un módulo del kernel

Si necesitas cargar un controlador de hardware (módulo) manualmente, puedes usar modprobe.

sudo modprobe nombre_modulo

Por ejemplo, para cargar el módulo del controlador de una tarjeta Wi-Fi:

sudo modprobe iwlwifi

Ejemplo 10: Descargar un módulo del kernel

Si quieres descargar un módulo (retirar el controlador de hardware):

sudo rmmod nombre_modulo

14. Gestión de Memoria de intercambio

Como vimos la memoria de intercambio (swap) en Linux es un espacio en disco utilizado como extensión de la memoria RAM cuando esta está llena. La gestión eficaz de swap es crucial para mantener el rendimiento del sistema cuando la RAM está sobrecargada. A continuación, te explico cómo gestionar el espacio de intercambio, desde su creación hasta su eliminación.

1. Conceptos Básicos

Memoria RAM: Memoria rápida que almacena datos y procesos activos.

Swap: Espacio en el disco utilizado para almacenar temporalmente datos que no caben en la RAM.

Intercambio (Swapping): Proceso de mover datos entre la RAM y el área de swap.

2. Verificar el Uso de Swap

Puedes consultar el uso actual de swap con varios comandos:

Ejemplo 1: Usar el comando free

El comando free muestra el uso de la memoria RAM y swap.

free -h

Salida esperada:

	total	used	free	shared	buff/cache	available
Mem:	7.8G	1.9G	4.5G	300M	1.4G	5.3G
Swap:	2.0G	0B	2.0G			

En este ejemplo, el sistema tiene 2.0 GB de swap disponible, pero no se está utilizando.

Ejemplo 2: Usar el archivo /proc/swaps

Este archivo contiene información sobre las particiones o archivos de swap en uso.

cat /proc/swaps

Salida esperada:

Filename	Type	Size	Used	Priority
/swapfile	file	2097148	0	-2

3. Crear un Archivo de Swap

Si necesitas más swap, puedes crear un archivo de swap adicional.

Pasos para crear un archivo de swap:

Crear el Archivo de Swap

Usa el comando dd para crear un archivo de swap. Por ejemplo, para crear un archivo de 1 GB:

sudo dd if=/dev/zero of=/swapfile bs=1M count=1024

Este comando crea un archivo llamado swapfile con 1 GB de tamaño.

Preparar el Archivo para el Swap

Usa el comando mkswap para preparar el archivo como espacio de intercambio:

sudo mkswap /swapfile

Activar el Archivo de Swap

Activa el archivo de swap con el comando swapon:

sudo swapon /swapfile

Verifica que el archivo de swap está activo usando el comando swapon --show o free -h.

Hacer el Swap Permanente

Para asegurar que el swap se active al reiniciar el sistema, añade el archivo al archivo /etc/fstab:

sudo nano /etc/fstab

Añade la siguiente línea al final del archivo:

/swapfile none swap sw 0 0

4. Eliminar un Archivo de Swap

Si ya no necesitas el archivo de intercambio, sigue estos pasos:

Desactivar el Archivo de Swap

Usa el comando swapoff para desactivar el archivo:

sudo swapoff /swapfile

Eliminar el Archivo de Swap

Elimina el archivo con el comando rm:

sudo rm /swapfile

Eliminar la Entrada de /etc/fstab

Si añadiste una entrada para el archivo de swap en /etc/fstab, elimínala para evitar errores en futuros arranques.

5. Gestionar el Uso de Swap con swappiness

El parámetro swappiness controla con qué frecuencia el sistema usa swap. El valor de swappiness varía de 0 a 100:

0: El sistema evita usar swap siempre que sea posible.

100: El sistema usa swap con más frecuencia, incluso si hay RAM disponible.

Verifica el valor actual de swappiness con:

cat /proc/sys/vm/swappiness

Cambia temporalmente el valor con:

sudo sysctl vm.swappiness=10

Para hacer el cambio permanente, edita el archivo /etc/sysctl.conf:

sudo nano /etc/sysctl.conf

Añade o modifica la línea:

vm.swappiness=10