

# Practica 1

## 1. Características de GNU/Linux:

- (a) Mencione y explique las características más relevantes de GNU/Linux.
- (b) Mencione otros sistemas operativos y compárelos con GNU/Linux en cuanto a los puntos mencionados en el inciso a.
- (c) ¿Qué es GNU?
- (d) Indique una breve historia sobre la evolución del proyecto GNU
- (e) Explique qué es la multitarea, e indique si GNU/Linux hace uso de ella.
- (f) ¿Qué es POSIX?

### 1.a\_ Características mas relevantes de GNU/linux.

- Es un Sistema Operativo tipo Unix (Unix like), pero libre → a diferencia de Unix, GNU/Linux es **libre** y **gratuito**, siguiendo la filosofía del software libre promovida por el proyecto GNU de Richard Stallman.
- S.O. diseñado por miles de programadores → GNU/Linux es el resultado de un esfuerzo colaborativo a nivel mundial. Miles de programadores y entusiastas contribuyen al desarrollo y mejora del sistema. Este modelo de desarrollo comunitario permite una evolución rápida y constante del software, con aportaciones en nuevas características, seguridad y eficiencia.
- S.O. gratuito y de libre distribución (se baja desde la Web, CD, etc.) → Uno de los grandes atractivos de GNU/Linux es que no tienes que pagar licencias para usarlo. Puedes descargar el sistema operativo y sus aplicaciones de forma gratuita desde Internet o mediante otros medios como CDs o USBs. Esto lo convierte en una opción accesible para todo tipo de usuarios.
- Existen diversas distribuciones (customizaciones) → GNU/Linux no es un único sistema operativo en el sentido tradicional, sino que es la base de **múltiples distribuciones**. Cada distribución (como Ubuntu, Fedora, Debian, Arch, etc.) está personalizada para satisfacer diferentes necesidades, desde usuarios domésticos hasta servidores empresariales. Esta variedad permite elegir la que mejor se adapte a tus requerimientos.
- Es código abierto, lo que nos permite estudiarlo, personalizarlo, auditarlo, aprovecharnos de la documentación, etc... → El hecho de que GNU/Linux sea **open source** significa que cualquiera puede acceder a su código fuente. Esto permite no solo aprender cómo funciona el sistema, sino también personalizarlo para adaptarlo a

necesidades específicas.

b\_

Característica	<b>GNU/Linux</b>	<b>Windows</b>	<b>macOS</b>	<b>FreeBSD</b>	<b>Android</b>
<b>Tipo Unix</b>	Sí (Unix-like)	No	Sí (Unix-based)	Sí (Unix-like)	Sí (basado en Linux)
<b>Desarrolladores</b>	Miles de programadores, comunidad global	Desarrollado por Microsoft	Desarrollado principalmente por Apple	Desarrollado por comunidad, pero más centralizado	Desarrollado por Google y fabricantes
<b>Costo</b>	Gratuito y libre	De pago, requiere licencia	Ligado a hardware Apple (caro)	Gratuito	Gratuito (pero versiones comerciales incluyen software propietario)
<b>Distribuciones</b>	Muchas distribuciones	No, pero tiene versiones (Home, Pro, etc.)	No tiene distribuciones, solo versiones	No tiene tantas distribuciones	Muchas variantes de Android
<b>Código abierto</b>	Totalmente open source	No (propietario)	Parcialmente open source, mayormente cerrado	Open source	Parcialmente open source, depende del fabricante

En resumen, **GNU/Linux** destaca frente a otros sistemas operativos por su naturaleza de código abierto, la posibilidad de múltiples distribuciones y su coste gratuito. Mientras que sistemas como **Windows** y **macOS** ofrecen entornos más cerrados y controlados, GNU/Linux y **FreeBSD** permiten una mayor personalización y flexibilidad, siendo excelentes opciones para usuarios avanzados y aquellos que buscan control total sobre su sistema.

c\_ **GNU** es un proyecto de software libre que fue iniciado por **Richard Stallman** en 1983 con el objetivo de crear un sistema operativo completamente libre, compatible con Unix, pero que respetara las libertades de los usuarios. GNU es un acrónimo que significa "**GNU's Not Unix**", lo que refleja su intención de ser similar a Unix en su funcionamiento, pero libre de las restricciones de licencias privativas.

d\_ Fue impulsado por Richard Stallman en el año 1983 con el fin de crear Unix libre (GNU). Para asegurar que el mismo fuera libre se necesito crear un marco regulatorio conocido como GPL (General Public License de GNU). En 1985, Stallman crea la FSF (Free Software Foundation),

con el fin de financiar el proyecto GNU. En 1990, GNU ya contaba con un editor de textos (Emacs), un compilador (GCC) y gran cantidad de bibliotecas que componen un Unix típico, faltaba el componente principal → El Nucleo (Kernel). En un principio se venía trabajando con un núcleo llamado TRIX, pero, en 1988 decidieron abandonarlo debido a que requería hardware muy costoso. En este momento se decidió adoptar como base el núcleo MACH para crear GNU Hurd, el cual tampoco prosperó. Linus Torvalds ya venía trabajando desde 1991 en un Kernel denominado Linux, el cual se distribuiría bajo licencia GPL. En el año 1992, Torvalds y Stallman deciden fusionar ambos proyectos, y es allí donde nace GNU/Linux.

e\_ La **multitarea** es una característica de los sistemas operativos que permite ejecutar **múltiples procesos** o tareas **simultáneamente**. Esto significa que varios programas o partes de un programa pueden estar funcionando al mismo tiempo, compartiendo los recursos del sistema como la CPU, la memoria y los dispositivos de entrada/salida.

▼ ¿GNU/Linux utiliza multitarea?

Sí, GNU/Linux **hace uso de la multitarea**, específicamente de la **multitarea por apropiación (preemptiva)**. Esto le permite manejar múltiples procesos al mismo tiempo de manera eficiente y estable.

f\_ **POSIX** (Portable Operating System Interface) es un conjunto de **normas y estándares** creados por el **IEEE** (Institute of Electrical and Electronics Engineers) para definir una interfaz estándar de programación entre los sistemas operativos tipo **Unix** y sus aplicaciones. El objetivo principal de POSIX es garantizar la **compatibilidad y portabilidad** de los programas entre diferentes sistemas operativos que sigan estos estándares, permitiendo que el software desarrollado en uno pueda ejecutarse en otro sin cambios significativos.

## 2. Distribuciones de GNU/Linux:

- (a) ¿Qué es una distribución de GNU/Linux? Nombre al menos 4 distribuciones de GNU/Linux y cite diferencias básicas entre ellas.
- (b) ¿En qué se diferencia una distribución de otra?
- (c) ¿Qué es Debian? Acceda al sitio e indique cuáles son los objetivos del proyecto y una breve cronología del mismo

2.a\_ Una **distribución de GNU/Linux** (o "distro") es una versión del sistema operativo que combina el **núcleo Linux** con una colección de **software, bibliotecas y herramientas** del proyecto **GNU** y otros proyectos. Cada distribución está diseñada para cumplir con diferentes propósitos y necesidades de los usuarios, ofreciendo un sistema operativo completo y listo para su uso. Las distribuciones varían en cuanto a la selección de programas, gestión de

paquetes, entorno de escritorio y herramientas administrativas, pero todas comparten el mismo núcleo Linux. Por ejemplo Ubuntu, Fedora, Debian, Arch Linux.

Distribución	Base	Gestor de paquetes	Enfoque principal	Entorno de escritorio predeterminado
Ubuntu	Debian	APT (.deb)	Facilidad de uso, usuarios nuevos	GNOME (por defecto)
Fedora	Independiente	DNF (.rpm)	Desarrolladores, últimas tecnologías	GNOME (por defecto)
Arch Linux	Independiente	Pacman	Configuración avanzada, minimalismo	Ninguno (el usuario elige)
Debian	Independiente	APT (.deb)	Estabilidad y fiabilidad	Múltiples opciones (GNOME, KDE, etc.)

b\_ pueden diferenciarse entre sí en varios aspectos clave, que afectan la experiencia de uso y el tipo de público al que están dirigidas. A continuación se explican las principales diferencias:

### 1. Base de la distribución:

- Algunas distribuciones están **basadas en otras** ya existentes, mientras que otras son **independientes**.
  - Ejemplo: **Ubuntu** está basada en **Debian**, mientras que **Arch Linux** es independiente.
  - Esto afecta la estructura interna, los gestores de paquetes y el ciclo de desarrollo.

### 2. Gestor de paquetes:

- Cada distribución usa un sistema de gestión de paquetes diferente para instalar, actualizar y desinstalar software. Algunos gestores de paquetes comunes son:
  - **APT** (Debian, Ubuntu): Utiliza paquetes en formato **.deb**.
  - **DNF** (Fedora): Utiliza paquetes en formato **.rpm**.
  - **Pacman** (Arch Linux): Formato de paquetes propio.
- La diferencia en el gestor de paquetes afecta la facilidad de uso y la compatibilidad de software.

### 3. Modelo de lanzamiento:

- **Lanzamientos fijos (Fixed Release)**: Algunas distribuciones lanzan versiones periódicamente (cada 6 meses, 1 año, etc.). Ejemplo: **Ubuntu**, **Debian**.
  - Ventajas: Estabilidad y menos interrupciones.
  - Desventajas: Software puede quedar desactualizado entre versiones.

- **Rolling Release:** Actualización continua en lugar de lanzamientos puntuales. Ejemplo: **Arch Linux**.
  - Ventajas: Siempre se tiene el software más reciente.
  - Desventajas: Riesgo de inestabilidad si las actualizaciones no son completamente probadas.

#### 4. Enfoque del usuario:

- Algunas distribuciones están diseñadas para usuarios **novatos** que buscan un sistema fácil de usar y amigable, mientras que otras están orientadas a **usuarios avanzados** que prefieren más control sobre el sistema.
  - Ejemplo: **Ubuntu** es amigable para principiantes, mientras que **Arch Linux** requiere mayor conocimiento técnico.

#### 5. Entorno de escritorio:

- Las distribuciones pueden ofrecer diferentes **entornos de escritorio** como predeterminados, que cambian la apariencia y la interacción con el sistema.
  - **GNOME** (Ubuntu, Fedora): Más moderno y minimalista.
  - **KDE Plasma** (Kubuntu): Personalizable y con un enfoque más clásico.
  - **XFCE** (Xubuntu): Ligero y orientado a la eficiencia de recursos.
- Algunas distribuciones permiten elegir entre varios entornos, mientras que otras prefieren uno específico.

#### 6. Estabilidad vs. Últimas tecnologías:

- Algunas distribuciones priorizan la **estabilidad**, como **Debian** o **CentOS**, que prefieren incluir software bien probado aunque no sea el más reciente.
- Otras, como **Fedora** o **Arch Linux**, están orientadas a usuarios que desean tener acceso a las **últimas tecnologías**, como nuevos entornos de escritorio, bibliotecas y herramientas de desarrollo.

#### 7. Soporte y comunidad:

- Algunas distribuciones tienen una gran comunidad de soporte detrás y son apoyadas por empresas, como **Ubuntu** (Canonical) o **Fedora** (Red Hat).
- Otras son mantenidas principalmente por comunidades más pequeñas o grupos de usuarios, como **Arch Linux** o **Debian**.
- El nivel de soporte técnico (documentación, foros, tutoriales) y la accesibilidad a actualizaciones también varía.

## 8. Propósito:

- Algunas distribuciones están orientadas a **usos específicos**:
  - **Kali Linux**: Para pruebas de penetración y seguridad.
  - **Ubuntu Server** o **CentOS**: Para servidores.
  - **Linux Mint**: Para usuarios de escritorio con una interfaz amigable y soporte multimedia.

c\_ **Debian GNU/Linux** es una distribución Linux de software libre, desarrollado por miles de voluntarios de todo el mundo, que colaboran a través de Internet. Los objetivos del proyecto Debian incluyen proporcionar un sistema operativo totalmente compuesto de software libre y ser accesible para un amplio espectro de arquitecturas de hardware, lo que refuerza su propósito de estar disponible para cualquier usuario o desarrollador interesado. A lo largo de su historia, Debian ha sido la base para muchas otras distribuciones, como Ubuntu y Linux Mint, y sigue siendo un proyecto comunitario, no comercial, coordinado por voluntarios de todo el mundo.

### 3. Estructura de GNU/Linux:

- (a) Nombre cuales son los 3 componentes fundamentales de GNU/Linux.
- (b) Mencione y explique la estructura básica del Sistema Operativo GNU/Linux.

3.a\_ El nucleo (Kernel), Shell o interprete de comandos, sistemas de archivos.

3.b\_ La estructura basica del SO GNU/Linux es el kernel → elemento encargado de que el software y el hardware puedan relacionarse y sus funciones mas importantes son la administracion de memoria, CPU y la e/s.

### 4. Kernel:

- (a) ¿Qué es? Indique una breve reseña histórica acerca de la evolución del Kernel de GNU/Linux.
- (b) ¿Cuáles son sus funciones principales?
- (c) ¿Cuál es la versión actual? ¿Cómo se definía el esquema de versionado del Kernel en versiones anteriores a la 2.4? ¿Qué cambió en el versionado se impuso a partir de la versión 2.6?
- (d) ¿Es posible tener más de un Kernel de GNU/Linux instalado en la misma máquina?
- (e) ¿Dónde se encuentra ubicado dentro del File System?
- (f) ¿El Kernel de GNU/Linux es monolítico? Justifique.

4.a\_ Es el corazon del sistema operativo.

- En 1991 Linus Torvalds inicia la programacion de un Kernel
- en 1992 se combina su desarrollo con GNU
- en 1994 aparece la version 1.0 y continua su desarrollo
- En mayo de 1996 se decide adoptar a Tux como mascota oficial de Linux 🐧
- En julio de 1996 se lanza la version 2.0 y se define lanomenclatura de versionado. Se desarrollo hasta febrero de 2004 y termino con la 2.0.40
- En enero de 1999 se lanza la version 2.2, que provee mejoras de portabilidad entre otras y se desarrolla hasta febrero de 2004 terminando en la version 2.2.26
- En 2001 se lanza la version 2.4 y se deja de desarrollar a fines del 2010 con la 2.4.37.11

b\_ El **núcleo de Linux** gestiona el hardware de la computadora, los recursos del sistema (CPU, memoria, dispositivos de almacenamiento, etc.) y proporciona una interfaz para que las aplicaciones interactúen con estos recursos. Es responsable de tareas como la **gestión de procesos**, la **gestión de memoria**, el control de dispositivos y la **seguridad** del sistema.

c\_ La versión actual del kernel de Linux es **6.10**, lanzada el **14 de julio de 2024**. Este kernel tiene soporte de seguridad hasta el **14 de noviembre de 2025**. En cuanto al versionado, antes de la versión 2.4, el sistema de numeración del kernel seguía un esquema en el que los números impares representaban versiones de desarrollo, mientras que los números pares eran versiones estables. Por ejemplo, una versión como 2.3 sería de desarrollo, mientras que 2.4 sería estable. A partir de la versión 2.6, este sistema cambió para simplificar la numeración. Se eliminó la distinción entre números pares e impares, y desde entonces se utiliza un esquema continuo para todas las versiones, tanto estables como de desarrollo, facilitando un flujo constante de actualizaciones y mejoras sin este tipo de diferenciación.

d\_ Sí, es posible tener más de un kernel de GNU/Linux instalado en la misma máquina. De hecho, muchas distribuciones de Linux permiten esta flexibilidad para que los usuarios puedan tener múltiples versiones del kernel instaladas y elegir cuál usar al arrancar el sistema.

e\_ El kernel de GNU/Linux se encuentra ubicado en el directorio **/boot** dentro del sistema de archivos. Este directorio almacena todos los archivos necesarios para que el sistema operativo pueda iniciar correctamente.

f\_ Sí, el **kernel de GNU/Linux** es considerado **monolítico**. Esto significa que la mayor parte de las funcionalidades esenciales del sistema operativo, como la gestión de memoria, manejo de procesos, y control de dispositivos, están integradas directamente en el núcleo y se ejecutan en el mismo espacio de memoria (espacio de kernel).

5. Intérprete de comandos (Shell):

- (a) ¿Qué es?
- (b) ¿Cuáles son sus funciones?
- (c) Mencione al menos 3 intérpretes de comandos que posee GNU/Linux y compárelos entre ellos.
- (d) ¿Dónde se ubican (path) los comandos propios y externos al Shell?
- (e) ¿Por qué considera que el Shell no es parte del Kernel de GNU/Linux?
- (f) ¿Es posible definir un intérprete de comandos distinto para cada usuario? ¿Desde dónde se define? ¿Cualquier usuario puede realizar dicha tarea?

5a\_ Un intérprete de comandos es un programa que recibe comandos del usuario, los interpreta y luego los ejecuta. Los usuarios escriben comandos en una interfaz de línea de comandos (CLI), y el intérprete de comandos traduce esos comandos en instrucciones que el sistema operativo puede entender y ejecutar. Los intérpretes de comandos son fundamentales en sistemas Unix, Linux y otros sistemas operativos similares, aunque también existen en otros sistemas como Windows.

b\_ Las funciones principales de un intérprete de comandos incluyen:

1. **Ejecutar Comandos:** Toma comandos introducidos por el usuario y los ejecuta. Estos comandos pueden incluir programas, scripts y utilidades del sistema.
2. **Interfaz de Usuario:** Proporciona una interfaz para que los usuarios interactúen con el sistema operativo. A través de esta interfaz, los usuarios pueden ejecutar aplicaciones, gestionar archivos y realizar otras tareas.
3. **Procesamiento de Comandos:** Interpreta y procesa los comandos escritos por el usuario. Esto incluye la expansión de variables, la interpretación de metacaracteres (como `*` para coincidencias de patrones) y la ejecución de comandos compuestos.
4. **Gestión de Tareas:** Permite la ejecución de múltiples comandos o tareas en paralelo mediante el uso de operadores como `&` para ejecutar en segundo plano o `&&` para ejecutar comandos condicionalmente.
5. **Redirección de Entrada y Salida:** Facilita la redirección de entrada y salida de los comandos mediante operadores como `>` para redirigir la salida a un archivo o `<` para leer la entrada desde un archivo.



6. **Pipelines:** Permite la creación de pipelines mediante el uso del operador `|`, que pasa la salida de un comando como entrada a otro comando.
7. **Script y Automatización:** Permite escribir scripts, que son secuencias de comandos que pueden automatizar tareas repetitivas o complejas.
8. **Manejo de Variables:** Soporta la creación y uso de variables que pueden almacenar información, como resultados de comandos o datos de configuración.
9. **Control de Flujo:** Soporta estructuras de control de flujo como bucles ( `for` , `while` ) y condicionales ( `if` , `else` ) dentro de scripts, permitiendo la creación de scripts más complejos.
10. **Configuración y Personalización:** Permite a los usuarios personalizar su entorno de trabajo mediante la configuración de archivos de perfil, alias y funciones personalizadas.

c\_ Los mas conocidos son Bash (Bourne Again Shell), Zsh (Z Shell), Fish (Friendly Interactive Shell).

- **Bash** es conocido por su estabilidad y amplia compatibilidad con scripts, siendo la opción más tradicional y ampliamente soportada.
- **Zsh** ofrece características avanzadas de autocompletado y personalización, siendo muy apreciado por usuarios que desean una mayor flexibilidad y un entorno altamente personalizable.
- **Fish** destaca por su facilidad de uso, interfaz amigable y autocompletado avanzado, aunque puede ser menos compatible con scripts de otros shells.

d\_

## 1. Comandos Internos al Shell

Los comandos internos, también conocidos como built-ins, están integrados directamente en el shell y no se encuentran en archivos ejecutables separados. Estos comandos son parte del propio intérprete de comandos y se ejecutan directamente desde la memoria del shell.

Ejemplos de comandos internos son `cd` , `echo` , `pwd` , y `exit` .

- **Ubicación:** No tienen una ubicación en el sistema de archivos porque son parte del binario del shell. Por ejemplo, el comando `cd` es un built-in de Bash, Zsh y otros shells, y su funcionalidad está implementada dentro del código del shell mismo.

## 2. Comandos Externos al Shell

Los comandos externos son archivos ejecutables que se encuentran en el sistema de archivos y que el shell puede invocar. Estos comandos pueden ser aplicaciones del sistema, utilidades, o

scripts. La ubicación de estos comandos puede variar, pero generalmente se encuentran en directorios específicos que están incluidos en la variable de entorno `PATH`.

## Ubicaciones Comunes de Comandos Externos:

### 1. `/bin`

- Contiene comandos esenciales que se necesitan para el funcionamiento básico del sistema y para el mantenimiento en modo de usuario único. Ejemplos incluyen `ls`, `cp`, y `mv`.

### 2. `/sbin`

- Almacena comandos de administración del sistema que generalmente son utilizados por el usuario root. Ejemplos incluyen `shutdown`, `ifconfig`, y `fdisk`.

### 3. `/usr/bin`

- Contiene la mayoría de los comandos disponibles para los usuarios. Este directorio es una de las ubicaciones principales para programas de usuario. Ejemplos incluyen `grep`, `awk`, y `curl`.

### 4. `/usr/sbin`

- Similar a `/sbin`, pero con comandos de administración que no son esenciales para el arranque del sistema. Ejemplos incluyen `apache2`, `cron`, y `useradd`.

### 5. `/local/bin`

- Usado para almacenar programas instalados localmente por el usuario que no forman parte de la distribución del sistema operativo. Este directorio se utiliza para software instalado manualmente.

### 6. `/local/sbin`

- Similar a `/local/bin`, pero para comandos de administración instalados localmente.

### 7. `/opt/bin`

- A menudo utilizado para software de terceros o aplicaciones que se instalan en el sistema. Cada paquete de software puede tener su propio directorio en `/opt`.


### 8. `/usr/local/bin`

- Utilizado para programas que se instalan localmente en el sistema fuera del sistema de gestión de paquetes del sistema operativo. Es una ubicación común para software instalado desde el código fuente o binarios descargados.

e\_ El Shell no es parte del Kernel porque sus responsabilidades y funciones son distintas. El Kernel gestiona el hardware y proporciona servicios del sistema, mientras que el Shell ofrece

una interfaz para que los usuarios interactúen con el sistema a través de comandos y scripts. Ambos son componentes esenciales del sistema operativo, pero operan en diferentes niveles y cumplen roles diferentes.

f\_ Sí, es posible definir un intérprete de comandos (shell) diferente para cada usuario en un sistema GNU/Linux. Esto se puede hacer configurando el shell predeterminado para cada usuario en el sistema.

 Consultar

#### 6. Sistema de Archivos (File System):


- (a) ¿Qué es?
- (b) Mencione sistemas de archivos soportados por GNU/Linux.
- (c) ¿Es posible visualizar particiones del tipo FAT y NTFS en GNU/Linux?
- (d) ¿Cuál es la estructura básica de los File System en GNU/Linux? Mencione los directorios más importantes e indique qué tipo de información se encuentra en ellos. ¿A qué hace referencia la sigla FHS?


6.a\_ Un sistema de archivos (file system en inglés) es una estructura que un sistema operativo utiliza para organizar y gestionar los archivos en un dispositivo de almacenamiento, como un disco duro, una unidad flash o un SSD. Su propósito principal es proporcionar una forma eficiente y organizada de almacenar, recuperar, y gestionar los datos.

b\_ GNU/Linux es compatible con una variedad de sistemas de archivos, tanto nativos como de otros sistemas operativos. Algunos de los más comunes incluyen:

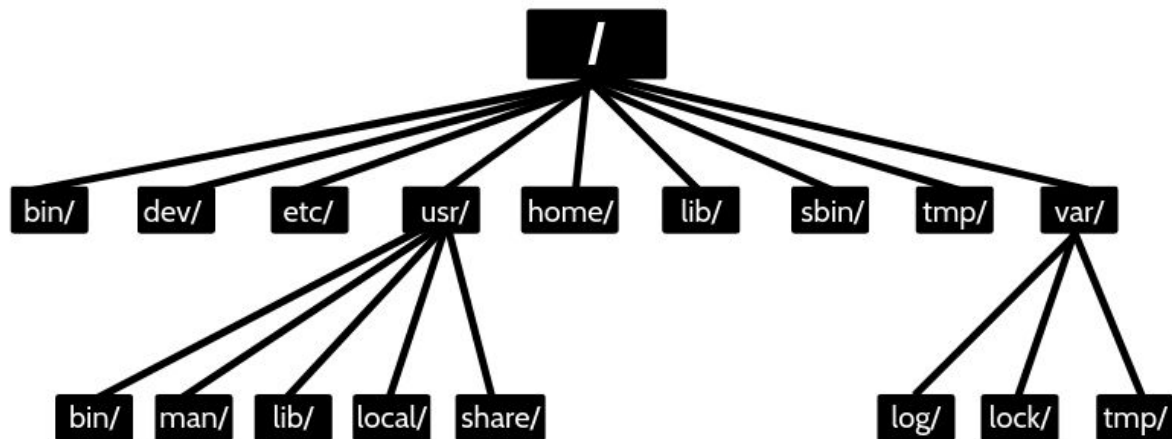
ext4, ext3, ext2, Btrfs, ReiserFS, FAT32, exFAT, NTFS, UFS, ZFS, JFS.

c\_ Sí, es posible visualizar y trabajar con particiones de tipo FAT y NTFS en GNU/Linux

 consultar

d\_ En GNU/Linux, la estructura básica de los sistemas de archivos se organiza de manera jerárquica, con un punto de partida común llamado "root" (). Desde este punto, el sistema de

archivos se ramifica en una serie de directorios estándar que organizan diferentes tipos de información y archivos del sistema. Esta estructura está definida por el estándar **FHS** (Filesystem Hierarchy Standard), que establece una estructura coherente para los directorios en sistemas Linux y UNIX.



- **/** (raíz): El directorio raíz que contiene todos los demás directorios del sistema.
- **/bin/**: Contiene los binarios ejecutables esenciales del sistema, como comandos básicos que necesitan estar disponibles para todos los usuarios.
- **/dev/**: Contiene archivos especiales que representan dispositivos del sistema, como discos duros, terminales, etc.
- **/etc/**: Almacena archivos de configuración de todo el sistema y scripts de inicio.
- **/usr/**: Contiene archivos de usuario, incluyendo aplicaciones y bibliotecas. Dentro de **/usr/**:
  - **/usr/bin/**: Contiene binarios y programas para los usuarios.
  - **/usr/man/**: Almacena las páginas de manuales de ayuda (man pages).
  - **/usr/lib/**: Contiene bibliotecas compartidas para los programas de usuario.
  - **/usr/local/**: Aquí se almacenan programas que el usuario instala de manera manual y que no forman parte de la distribución estándar.
  - **/usr/share/**: Almacena datos compartidos entre los programas, como archivos de configuración de ejemplo o datos comunes.
- **/home/**: Aquí se encuentran los directorios de los usuarios del sistema. Cada usuario tiene su propio subdirectorio dentro de **/home/**.

- **/lib/**: Contiene las bibliotecas compartidas del sistema, necesarias para que funcionen los binarios en `/bin/` y `/sbin/`.
- **/sbin/**: Contiene binarios esenciales para el sistema que solo puede ejecutar el superusuario (root), como herramientas de administración del sistema.
- **/tmp/**: Directorio temporal que es utilizado para almacenar archivos temporales creados por aplicaciones o el sistema. El contenido de este directorio suele ser eliminado al reiniciar el sistema.
- **/var/**: Almacena datos variables que pueden cambiar durante el funcionamiento del sistema, como logs, archivos de bloqueo y temporales.
  - **/var/log/**: Contiene archivos de registro del sistema.
  - **/var/lock/**: Almacena archivos de bloqueo usados para indicar que un recurso del sistema está siendo utilizado.
  - **/var/tmp/**: Almacena archivos temporales que persisten entre reinicios.

## FHS (Filesystem Hierarchy Standard)

La **sigla FHS** hace referencia al **Filesystem Hierarchy Standard**, que es el estándar que define la estructura y el contenido esperado de los directorios en sistemas operativos similares a Unix, como GNU/Linux. El objetivo del FHS es facilitar la interoperabilidad, asegurando que los sistemas de archivos tengan una estructura coherente y predecible para los administradores y usuarios.

### 7. Particiones:

- Definición. Tipos de particiones. Ventajas y Desventajas.
- ¿Cómo se identifican las particiones en GNU/Linux? (Considere discos IDE, SCSI y SATA).
- ¿Cuántas particiones son necesarias como mínimo para instalar GNU/Linux? Nómbralas indicando tipo de partición, identificación, tipo de File System y punto de montaje.
- Ejemplifique diversos casos de particionamiento dependiendo del tipo de tarea que se deba realizar en su sistema operativo.
- ¿Qué tipo de software para particionar existe? Menciónelos y compare.

7.a\_ Una **partición** es una división lógica de un disco duro físico, que permite que un solo disco pueda albergar diferentes sistemas de archivos o sistemas operativos. A través de particiones, un disco duro se puede dividir en áreas separadas, cada una con su propio sistema de archivos y propósito. Las particiones permiten organizar mejor los datos, mejorar el rendimiento, facilitar la administración y habilitar la coexistencia de varios sistemas operativos en una misma máquina.

## Tipos de Particiones

En un esquema clásico de partición (usando la tabla de particiones MBR, que es uno de los métodos tradicionales), se pueden encontrar los siguientes tipos de particiones:

### 1. Partición Primaria:

- Es una partición directa en el disco físico.
- Un disco puede tener hasta **cuatro particiones primarias**.
- Solo una partición primaria puede estar activa a la vez en el arranque, donde se instala el sistema operativo.

### 2. Partición Extendida:

- Es una partición especial que actúa como un contenedor de otras particiones llamadas **particiones lógicas**.
- Solo puede existir una partición extendida por disco, pero dentro de ella se pueden crear muchas particiones lógicas.
- No puede contener datos directamente, solo particiones lógicas.

### 3. Particiones Lógicas:

- Son las particiones dentro de la partición extendida.
- Se pueden crear tantas particiones lógicas como el espacio en disco lo permita.
- Normalmente se usan para datos o para instalar sistemas adicionales, aunque no pueden arrancar directamente como una partición primaria.

Con la llegada del esquema de particiones **GPT (GUID Partition Table)**, que reemplaza a MBR en discos más grandes y sistemas modernos, las restricciones cambian:

### 1. GPT (GUID Partition Table):

- Permite crear hasta **128 particiones** en un solo disco.
- No distingue entre particiones primarias, extendidas o lógicas, ya que todas son equivalentes.
- Es compatible con discos de más de 2 TB (algo que MBR no soporta).

## Ventajas y Desventajas Generales de las Particiones

### Ventajas de Particionar un Disco:

- **Organización:** Facilita la separación de sistemas operativos, datos y archivos de configuración en distintas áreas del disco.
- **Seguridad:** Si un sistema de archivos se corrompe, el daño se limita a una partición específica.

- **Rendimiento:** Puede mejorar el rendimiento del sistema, ya que cada partición puede tener su propio sistema de archivos optimizado.
- **Mantenimiento:** Simplifica la reinstalación de sistemas operativos, ya que se puede formatear la partición del sistema sin afectar los datos en otras particiones.

#### Desventajas de Particionar un Disco:

- **Espacio desaprovechado:** Si no se planifica correctamente, algunas particiones pueden quedarse sin espacio mientras otras tienen espacio de sobra.
- **Complejidad:** Puede hacer que la administración del disco sea más complicada, especialmente si se manejan muchos sistemas operativos o particiones lógicas.
- **Riesgo de borrado accidental:** La gestión incorrecta de particiones puede llevar a la pérdida de datos o a la sobrescritura de particiones importantes.

En resumen, las particiones permiten una gestión más eficiente y segura de los discos, pero su uso requiere una planificación adecuada para evitar problemas de espacio o pérdida de datos.

b\_ En **GNU/Linux**, las particiones se identifican a través de archivos de dispositivos que se encuentran en el directorio `/dev`. Los discos y particiones tienen nombres específicos dependiendo del tipo de disco y del controlador usado (IDE, SCSI, SATA).

## Identificación de Discos y Particiones

### 1. Discos IDE (Integrated Drive Electronics):

- En sistemas más antiguos, los discos **IDE** se identificaban con el prefijo `hd` (de **hard drive**).
- Los nombres de los dispositivos IDE en **GNU/Linux** siguen la siguiente convención:
  - `hda`: Primer disco IDE (generalmente maestro en el canal IDE 1).
  - `hdb`: Segundo disco IDE (generalmente esclavo en el canal IDE 1).
  - `hdc`: Tercer disco IDE (generalmente maestro en el canal IDE 2).
  - `hdd`: Cuarto disco IDE (generalmente esclavo en el canal IDE 2).
- Las **particiones** en estos discos se identifican añadiendo un número al final:
  - Ejemplo: `hda1` (primera partición del primer disco IDE), `hdb2` (segunda partición del segundo disco IDE).

### 2. Discos SCSI (Small Computer System Interface), SATA (Serial ATA) y USB:

- Los discos **SCSI**, **SATA** y **USB** utilizan el prefijo `sd` (de **SCSI disk**, aunque se usa también para SATA y USB).
- Los nombres de los dispositivos SCSI, SATA y USB siguen esta convención:

- `sda` : Primer disco SCSI, SATA o USB.
- `sdb` : Segundo disco.
- `sdc` : Tercer disco, y así sucesivamente.
- Las **particiones** se identifican de manera similar, agregando un número al final:
  - Ejemplo: `sda1` (primera partición del primer disco), `sdb2` (segunda partición del segundo disco).

### 3. NVMe (Non-Volatile Memory Express):

- Los discos NVMe, que se usan para almacenamiento de alta velocidad, tienen un esquema de nombres diferente, comenzando con `nvme` :
  - `nvme0n1` : Primer dispositivo NVMe.
  - `nvme0n1p1` : Primera partición del primer dispositivo NVMe.
  - `nvme0n1p2` : Segunda partición del primer dispositivo NVMe.

c\_ Para instalar **GNU/Linux**, se necesitan como **mínimo dos particiones**. Estas son fundamentales para garantizar que el sistema operativo funcione correctamente. Sin embargo, en instalaciones más avanzadas o para mejorar la organización del sistema, es común utilizar más particiones.

## Mínimo Requerido para Instalar GNU/Linux

### 1. Partición de raíz ( `/` ):

- **Tipo de partición:** Primaria o lógica (dependiendo del esquema de particionado).
- **Identificación:** En un disco SATA o SCSI, se identifica como algo similar a `sda1` (la primera partición del primer disco).
- **Tipo de File System:** Comúnmente se usa **EXT4** (por su balance entre rendimiento y estabilidad), aunque también se pueden usar otros sistemas de archivos como **XFS**, **Btrfs**, o **EXT3**.
- **Punto de montaje:** `/` (directorio raíz).
- **Descripción:** Es la partición principal donde se instalan el sistema operativo y todas las aplicaciones.

### 2. Partición de intercambio (swap):

- **Tipo de partición:** Primaria o lógica.
- **Identificación:** Se identifica como `sda2`, `sdb2`, etc., dependiendo del disco y el orden de creación.



- **Tipo de File System:** No tiene un sistema de archivos tradicional, ya que es un espacio dedicado a la memoria virtual del sistema.
- **Punto de montaje:** No se monta en un directorio como tal, ya que es gestionada automáticamente por el sistema.
- **Descripción:** Swap es el espacio de intercambio que el sistema utiliza como memoria virtual cuando la memoria RAM está llena. Su tamaño recomendado varía según la cantidad de RAM disponible y el uso previsto del sistema.

d\_

## 1. Sistema de escritorio general (uso personal)

**Escenario:** Un usuario regular que utiliza el sistema para navegar por internet, crear documentos, ver videos y otras tareas cotidianas.

- **Particiones:**

1. **Partición `/` (raíz):**

- **Tamaño sugerido:** 20-30 GB.
- **Sistema de archivos:** EXT4.
- **Descripción:** Contiene el sistema operativo y las aplicaciones.

2. **Partición `/home`:**

- **Tamaño sugerido:** Resto del disco.
- **Sistema de archivos:** EXT4.
- **Descripción:** Almacena los archivos personales del usuario (documentos, videos, configuraciones personales). Facilita la reinstalación del sistema operativo sin afectar los datos personales.

3. **Partición de intercambio (swap):**

- **Tamaño sugerido:** Entre 2 GB y el doble de la memoria RAM.
- **Descripción:** Memoria virtual utilizada cuando la RAM se llena.

**Ventajas:**

- Separar `/home` facilita la actualización o reinstalación del sistema sin perder datos personales.
- Swap asegura que el sistema no se quede sin memoria en tareas pesadas.

## 2. Servidor web o servidor de base de datos

**Escenario:** Un servidor que aloja sitios web y bases de datos, necesita estabilidad y seguridad.

- **Particiones:**

1. **Partición `/` (raíz):**

- **Tamaño sugerido:** 10-20 GB.
- **Sistema de archivos:** EXT4 o XFS.
- **Descripción:** Almacena el sistema operativo y aplicaciones básicas.

2. **Partición `/var`:**

- **Tamaño sugerido:** 50 GB o más, dependiendo del tamaño esperado de los archivos de log y la base de datos.
- **Sistema de archivos:** EXT4 o XFS.
- **Descripción:** Los servidores web y las bases de datos almacenan archivos de log y datos en `/var`. Separar esta partición mejora la gestión y evita que los logs inunden el disco raíz.

3. **Partición `/srv` (opcional):**

- **Tamaño sugerido:** Depende del tamaño del sitio o servicio alojado.
- **Sistema de archivos:** EXT4 o XFS.
- **Descripción:** Almacena los datos del servicio web o base de datos (ej. archivos web, repositorios de datos). Facilita la organización de datos del servicio.

4. **Partición de intercambio (swap):**

- **Tamaño sugerido:** Entre 2 y 8 GB.

**Ventajas:**

- Mantener `/var` separado evita que los archivos de log o bases de datos ocupen todo el espacio de la partición raíz.
- `/srv` es ideal si se gestionan muchos servicios o sitios web.

e\_

Herramienta	Tipo de interfaz	Soporte GPT	Redimensionar sin pérdida de datos	Ideal para	Nivel de dificultad
<b>GNU Parted</b>	Línea de comandos	Sí	Sí	Usuarios avanzados	Alta
<b>GParted</b>	Gráfica	Sí	Sí	Usuarios intermedios	Baja

<b>fdisk</b>	Línea de comandos	Sí (reciente)	No	Tareas básicas	Media
<b>cfdisk</b>	Menú en terminal	Sí (reciente)	No	Tareas básicas	Baja
<b>KDE Partition Manager</b>	Gráfica	Sí	Sí	Usuarios KDE	Baja
<b>LVM</b>	Línea de comandos	N/A	Sí (con volúmenes lógicos)	Servidores	Alta
<b>GNOME Disks Utility</b>	Gráfica	Sí	Sí (limitado)	Usuarios GNOME	Baja

- **GParted** es la opción más equilibrada para la mayoría de los usuarios que necesitan particionar discos sin perder datos, gracias a su interfaz gráfica y soporte amplio.
- **GNU Parted** y **LVM** son ideales para usuarios avanzados o administradores de sistemas que necesitan herramientas potentes y flexibles.
- **fdisk** y **cfdisk** son opciones ligeras y rápidas, pero limitadas en características avanzadas como el redimensionamiento sin pérdida de datos.
- **KDE Partition Manager** y **GNOME Disks Utility** son perfectas para usuarios de entornos gráficos KDE o GNOME que prefieren herramientas gráficas básicas.

#### 1. Arranque (bootstrap) de un Sistema Operativo:

- ¿Qué es el BIOS? ¿Qué tarea realiza?
- ¿Qué es UEFI? ¿Cuál es su función?
- ¿Qué es el MBR? ¿Que es el MBC?
- ¿A qué hacen referencia las siglas GPT? ¿Qué sustituye? Indique cuál es su formato.
- ¿Cuál es la funcionalidad de un "Gestor de Arranque"? ¿Qué tipos existen? ¿Dónde se instalan? Cite gestores de arranque conocidos.
- ¿Cuáles son los pasos que se suceden desde que se prende una computadora hasta que el Sistema Operativo es cargado (proceso de bootstrap)?
- Analice el proceso de arranque en GNU/Linux y describa sus principales pasos.
- ¿Cuáles son los pasos que se suceden en el proceso de parada (shutdown) de GNU/Linux?
- ¿Es posible tener en una PC GNU/Linux y otro Sistema Operativo instalado? Justifique.

1.a\_ El BIOS es un software de bajo nivel almacenado en un chip en la placa base de un sistema. Su función principal es iniciar y configurar los componentes básicos del hardware cuando enciendes la computadora. Se ejecuta antes de cargar el sistema operativo y proporciona una interfaz entre el hardware y el software.

## Tareas principales del BIOS:

1. **POST (Power-On Self Test):** Al encender la computadora, el BIOS realiza una verificación básica del hardware (RAM, teclado, unidades de almacenamiento, etc.) para asegurarse de que todo funcione correctamente.
2. **Cargar el gestor de arranque:** Después de verificar el hardware, el BIOS busca un dispositivo de arranque, como un disco duro o una unidad USB, donde esté almacenado el sistema operativo. Luego, transfiere el control al gestor de arranque (como GRUB en Linux).
3. **Configuración del hardware:** El BIOS configura los dispositivos y parámetros de hardware como la frecuencia de la CPU, las configuraciones de la memoria, los dispositivos de almacenamiento y los periféricos.
4. **Interfaz de configuración:** Permite a los usuarios modificar ciertos parámetros del sistema, como el orden de arranque, habilitar o deshabilitar periféricos, y ajustar otras configuraciones relacionadas con el hardware.

b\_ UEFI (Unified Extensible Firmware Interface) es una especificación que define una interfaz entre el sistema operativo y el firmware de la placa base del equipo. Se diseñó como una evolución del BIOS (Basic Input/Output System), que es el sistema básico de entrada y salida tradicional utilizado para arrancar un sistema operativo en los equipos. La función principal de UEFI es **iniciar el sistema operativo** en un equipo, proporcionando una interfaz entre el firmware del hardware y el software del sistema operativo. UEFI incluye la función de **Secure Boot**, que verifica la autenticidad del software que se ejecuta durante el arranque, como el sistema operativo y los controladores. Esto ayuda a prevenir que software malintencionado (como rootkits o bootkits) se ejecute antes de que se cargue el sistema operativo.

c\_ El **MBR** es un estándar de particionado y el sector de arranque que se encuentra al inicio de un disco duro (sector 0). Fue introducido en los años 80 y ha sido utilizado tradicionalmente en sistemas BIOS para arrancar sistemas operativos.

### Funciones principales del MBR:

- Contiene la tabla de particiones de un disco duro, lo que permite al sistema operativo identificar y acceder a las particiones.
- Almacena el código de arranque (boot loader) que se ejecuta para cargar el sistema operativo desde la partición activa.
- Solo soporta discos de hasta 2 TB y permite un máximo de cuatro particiones primarias.

### MBC (Master Boot Code)

El **MBC** es el código específico que se almacena dentro del MBR y es responsable de iniciar el proceso de arranque del sistema operativo. Es la parte del MBR que contiene las instrucciones

que el BIOS (o UEFI en modo de compatibilidad) ejecuta para localizar y cargar el gestor de arranque en la memoria.

En resumen:

- El **MBR** es el sector de arranque y la estructura que define las particiones en un disco.
- El **MBC** es el código de arranque almacenado dentro del MBR encargado de iniciar el arranque del sistema operativo.

d\_ Las siglas **GPT** hacen referencia a **GUID Partition Table** (Tabla de Particiones GUID). Es un estándar de particionado utilizado en discos duros y otros medios de almacenamiento.

### ¿Qué sustituye?

GPT reemplaza al antiguo esquema de particionado **MBR** (Master Boot Record), superando sus limitaciones, como el soporte de discos de hasta 2 TB y el límite de cuatro particiones primarias.

### Formato de GPT:

- **GUID (Globally Unique Identifier):** GPT utiliza identificadores únicos globales (GUID) para identificar cada partición, lo que mejora la organización y gestión de las particiones.
- **Soporte para discos grandes:** GPT permite el uso de discos duros mucho más grandes (hasta 9.4 zettabytes).
- **Número de particiones:** A diferencia del MBR, que solo permite cuatro particiones primarias, GPT permite crear hasta 128 particiones sin necesidad de particiones extendidas.
- **Respaldo del encabezado GPT:** GPT almacena una copia de seguridad del encabezado de la tabla de particiones en el final del disco, lo que aumenta la seguridad y recuperación en caso de corrupción del encabezado principal.

En resumen, GPT es un esquema moderno y eficiente para particionar discos duros, diseñado para superar las limitaciones del MBR y proporcionar mayor flexibilidad y seguridad.

e\_ Un **gestor de arranque** (bootloader) es un programa cuya función principal es cargar el sistema operativo cuando se enciende el equipo. Actúa como un intermediario entre el firmware del sistema (BIOS o UEFI) y el sistema operativo, seleccionando qué sistema operativo cargar (en caso de tener varios) y dónde encontrarlo en el disco duro.

### Tipos de Gestores de Arranque:

1. **Gestores de arranque de un solo sistema:** Estos gestores simplemente cargan un único sistema operativo.

2. **Gestores de arranque múltiples (multiboot):** Permiten seleccionar entre varios sistemas operativos instalados en el equipo, facilitando la elección del usuario al arrancar el sistema.

### Dónde se Instalan:

- Generalmente, los gestores de arranque se instalan en el **sector de arranque del disco**:
  - **MBR** (Master Boot Record) en sistemas BIOS o UEFI en modo de compatibilidad.
  - **EFI System Partition (ESP)** en sistemas UEFI.

### Gestores de Arranque Conocidos:

GRUB, LILO, rEFInd, Windows Boot Manager, SYSLINUX/ISOLINUX

f\_ El proceso de arranque o **bootstrap** es el conjunto de pasos que ocurren desde que se enciende una computadora hasta que el sistema operativo está completamente cargado y listo para usarse. A continuación se mencionan estos pasos:

- Encendido → POST (BIOS/UEFI)
- Selección del dispositivo de arranque
- Carga del gestor de arranque
- Carga del kernel del sistema operativo
- Inicialización de servicios y procesos
- Carga del entorno de usuario (gráfico o CLI)

g\_ Proceso de arranque de GNU/Lin

- **Ejecución del firmware (BIOS/UEFI)** → Selección del dispositivo de arranque.
- **Carga del gestor de arranque** (GRUB u otro).
- **Carga del kernel de Linux.**
- **Inicialización del initramfs.**
- **Montaje del sistema de archivos raíz.**
- **Ejecución del sistema init** (systemd o SysVinit).
- **Inicialización de servicios y scripts de inicio.**
- **Carga del entorno de usuario** (gráfico o CLI).

h\_ El proceso de **parada** o **shutdown** en GNU/Linux implica una serie de pasos para asegurar que el sistema se apague de manera segura, guardando datos y liberando recursos. A continuación se mencionan los pasos principales que se suceden durante este proceso:

- **Recepción de la señal de apagado:** Comando o botón de encendido.
- **Notificación a los servicios:** Envío de señales de terminación.
- **Desmontaje de sistemas de archivos:** Liberación de recursos de almacenamiento.
- **Finalización de sesiones de usuario:** Cierre de aplicaciones y sesiones.
- **Parada de servicios y daemons:** Detención ordenada de todos los procesos.
- **Apagado del sistema de archivos:** Sincronización y aseguramiento de integridad de datos.
- **Apagado del hardware:** Desactivación de componentes físicos.
- **Reinicio o apagado completo:** Dependiendo del comando, reinicio o apagado final.

i\_ Si es posible, realizando o una particion del disco local, o con dos discos distintos. Tener en cuenta que primero debemos instalar windows y despues linux, si instalamos primero linux y despues windows, este, nos pisara por completo el SO.

## 9. Archivos:

- (a) ¿Cómo se identifican los archivos en GNU/Linux?
- (b) Investigue el funcionamiento de los editores vi y mcedit, y los comandos cat y more.
- (c) Cree un archivo llamado "prueba.exe" en su directorio personal usando el vi. El mismo debe contener su número de alumno y su nombre.
- (d) Investigue el funcionamiento del comando file. Pruébelo con diferentes archivos. ¿Qué diferencia nota?

### 9.a\_ Se identifican en:

- Ruta del archivo
- Nombre del archivo
- Inodo
- Permisos y propietario
- Tipo de archivo
- Ruta de acceso
- Metadatos

b\_

## Editor `vi`

`vi` es un editor de texto de pantalla completa que se encuentra en la mayoría de los sistemas Unix y Linux. Tiene dos modos principales de operación:

1. **Modo de Comando:** Este es el modo por defecto cuando se abre `vi`. En este modo, puedes moverte por el texto, eliminar, copiar y pegar contenido, y ejecutar comandos.
  - **Navegación:** Usa las teclas de dirección o las teclas `h`, `j`, `k`, `l` para moverte por el texto.
  - **Comandos:** Puedes usar comandos como `dd` para eliminar una línea, `yy` para copiar una línea y `p` para pegar.
  - **Guardar y salir:** Usa `:w` para guardar, `:q` para salir y `:wq` para guardar y salir.
2. **Modo de Inserción:** Este modo permite la edición del texto. Puedes ingresar al modo de inserción presionando `i` (insertar antes del cursor), `a` (insertar después del cursor), o `o` (abrir una nueva línea debajo del cursor).
  - **Volver al modo de comando:** Presiona `Esc` para regresar al modo de comando desde el modo de inserción.

## Editor `mcedit`

`mcedit` es un editor de texto basado en la interfaz de usuario de **Midnight Commander** (MC), que es un administrador de archivos de texto basado en texto. Es más amigable y proporciona una interfaz visual más fácil de usar comparado con `vi`.

## Comando `cat`

El comando `cat` (concatenate) se utiliza para mostrar el contenido de archivos, concatenar archivos y redirigir la salida a otros archivos.

## Comando `more`

El comando `more` se usa para visualizar el contenido de un archivo una pantalla a la vez. Es útil para archivos grandes donde se desea ver el contenido en partes.

d\_ El comando `file` en GNU/Linux se utiliza para identificar el tipo de archivo de manera que no se basa únicamente en la extensión del archivo, sino que examina el contenido del archivo para determinar su tipo. Esto lo hace al leer una serie de características específicas del archivo, como su formato y contenido interno.



10. Indique qué comando es necesario utilizar para realizar cada una de las siguientes acciones.

Investigue su funcionamiento y parámetros más importantes:

- (a) Cree la carpeta ISO2017
- (b) Acceda a la carpeta (cd)
- (c) Cree dos archivos con los nombres iso2017-1 e iso2017-2 (touch)
- (d) Liste el contenido del directorio actual (ls)
- (e) Visualizar la ruta donde estoy situado (pwd)
- (f) Busque todos los archivos en los que su nombre contiene la cadena "iso\*" (find)
- (g) Informar la cantidad de espacio libre en disco (df)
- (h) Verifique los usuarios conectados al sistema (who)
- (i) Acceder a el archivo iso2017-1 e ingresar Nombre y Apellido
- (j) Mostrar en pantalla las últimas líneas de un archivo (tail).

a. mkdir ISO2017

b. cd ISO2017

c. touch iso2017-1 iso2017-2

d. ls

e. pwd

f. find . -name 'iso\*'

g. df

h. who

i. nano iso2017-1

j. tail nombreArchivo

11. Investigue su funcionamiento y parámetros más importantes:

- (a) shutdown
- (b) reboot
- (c) halt
- (d) locate
- (e) uname
- (f) gmesg
- (g) lspci
- (h) at
- (i) netstat
- (j) mount

- (k) umount
- (l) head
- (m) losetup
- (n) write
- (ñ) mkfs
- (o) fdisk (con cuidado)

a. El comando `shutdown` se usa para apagar o reiniciar el sistema.

**Parámetros importantes:**

- `h` : Apaga el sistema (también se puede usar `-halt` ).
- `r` : Reinicia el sistema.
- `p` : Apaga el sistema y apaga la alimentación (dependiendo del hardware).
- `now` : Ejecuta el comando inmediatamente. Por ejemplo, `shutdown -h now` .
- `+m` : Programa el apagado para dentro de `m` minutos. Por ejemplo, `shutdown -r +10` reinicia en 10 minutos.
- `c` : Cancela un apagado programado.

b. El comando `reboot` reinicia el sistema de manera inmediata.

c. El comando `halt` detiene inmediatamente el sistema, apagando el hardware sin realizar un apagado limpio.

**Parámetros importantes:**

- `p` : Apaga el sistema después de detenerlo.

d. El comando `locate` se utiliza para encontrar archivos en el sistema basándose en una base de datos indexada.

**Parámetros importantes:**

- `i` : Ignora el caso al buscar. Por ejemplo, `locate -i archivo` .
- `r` : Utiliza una expresión regular para buscar. Por ejemplo, `locate -r 'patrón'` .

e. El comando `uname` muestra información sobre el sistema operativo y el kernel.

**Parámetros importantes:**

- `a` : Muestra toda la información disponible, como nombre del kernel, versión, hostname, etc.
- `r` : Muestra la versión del kernel.
- `s` : Muestra el nombre del sistema operativo.

- `m`: Muestra la arquitectura del hardware (por ejemplo, x86\_64).

f. El comando `dmesg` muestra los mensajes del buffer del kernel, que incluyen mensajes del sistema y de hardware.

**Parámetros importantes:**

- `T`: Convierte las marcas de tiempo en un formato legible para humanos.
- `H`: Formatea la salida para mejorar la legibilidad.

g. El comando `lspci` muestra información sobre todos los dispositivos PCI en el sistema.

**Parámetros importantes:**

- `v`: Muestra información detallada sobre cada dispositivo.
- `nn`: Muestra identificadores de dispositivo y proveedor en formato numérico.
- `k`: Muestra los módulos del kernel asociados a cada dispositivo.

h. El comando `at` programa la ejecución de comandos en una hora específica.

**Parámetros importantes:**

- `l`: Lista los trabajos programados.
- `d`: Elimina un trabajo programado.
- `f archivo`: Lee los comandos desde un archivo en lugar de desde la entrada estándar.

i. El comando `netstat` muestra información sobre las conexiones de red, tablas de enrutamiento y estadísticas de interfaz.

**Parámetros importantes:**

- `t`: Muestra conexiones TCP.
- `u`: Muestra conexiones UDP.
- `a`: Muestra todas las conexiones y puertos en escucha.
- `r`: Muestra la tabla de enrutamiento.

j. El comando `mount` se utiliza para montar sistemas de archivos.

**Parámetros importantes:**

- `t tipo`: Especifica el tipo de sistema de archivos (por ejemplo, `ext4`, `ntfs`).
- `o opciones`: Especifica opciones de montaje adicionales (por ejemplo, `ro` para solo lectura).

k. El comando `umount` se usa para desmontar sistemas de archivos.

l. El comando `head` muestra las primeras líneas de un archivo.

**Parámetros importantes:**

- `n número` : Especifica el número de líneas a mostrar. Por ejemplo, `head -n 20 archivo` muestra las primeras 20 líneas.
- `c número` : Especifica el número de bytes a mostrar.

m. El comando `losetup` se usa para configurar y gestionar dispositivos de loopback.

#### Parámetros importantes:

- `f` : Encuentra el primer dispositivo loopback libre.
- `d` : Desmonta un dispositivo loopback.
- `p` : Actualiza la información de particiones en el dispositivo loopback.

n. El comando `write` se usa para enviar mensajes a otros usuarios en el sistema.

#### Parámetros importantes:

- `usuario` : El nombre del usuario al que enviar el mensaje.
- `tty` : El terminal en el que el usuario está conectado (opcional).

ñ. El comando `mkfs` se utiliza para crear un sistema de archivos en un dispositivo de almacenamiento.

#### Parámetros importantes:

- `t tipo` : Especifica el tipo de sistema de archivos (por ejemplo, `ext4`, `xfs`).
- `n` : Realiza una prueba, no formatea realmente.

o. El comando `fdisk` se usa para manipular las tablas de particiones de discos duros.

#### Parámetros importantes:

- `l` : Lista las particiones de todos los discos.
- `u` : Muestra los tamaños de partición en sectores (en lugar de cilindros).
- `x` : Muestra detalles adicionales (modo experto).

12.

## Directorios Comunes para Comandos:

1. `/bin` :

Contiene los binarios esenciales que se necesitan para arrancar el sistema y para que funcionen en modo monousuario (modo de recuperación). Algunos comandos almacenados aquí incluyen:

- `uname`
- `shutdown`

- `reboot`
- `halt`
- `ls`

2. `/sbin` :

Contiene binarios de administración de sistema, normalmente usados por el superusuario (root). Aquí se encuentran comandos relacionados con la gestión del sistema y el hardware, como:

- `fdisk`
- `mkfs`
- `shutdown`
- `reboot`
- `halt`

3. `/usr/bin` :

Almacena la mayoría de los comandos que no son esenciales para el arranque del sistema pero que son utilizados comúnmente por los usuarios. Algunos de los comandos de esta ubicación son:

- `locate`
- `find`
- `write`
- `df`
- `tail`
- `head`
- `lspci`
- `netstat`

4. `/usr/sbin` :

Contiene comandos de administración de sistema adicionales que no son esenciales para el arranque. Aquí también se encuentran herramientas utilizadas por el administrador del sistema, como:

- `losetup`
- `mount`
- `umount`