Tp1

1: **Características de GNU/Linux:**

▼ **(a) Mencione y explique las características más relevantes de GNU/Linux.**

- Codigo libre/ comunidad grande atras: Al ser codigo libre, existen varios individuos que pueden hacer bifurcaciones

- Seguridad: Al ser publico el codigo, muchos usuarios pueden ver los cambios realizados y detectar posible código malicioso

- Personalizable: es altamente personalizable pudiendo elegir entre varios distribuciones

▼ **(b) Mencione otros sistemas operativos y compárelos con GNU/Linux en cuanto a los**
puntos mencionados en el inciso a.

Windows: no es de codigo abierto, a comparacion de linux tiene menor rendimiendo, pero tiene mas interfaz grafica para el usuario final(? tampoco es gratuito

MacOs: no es de codigo abierto y solo se ejecuta en dispositivos apple, tiene un grado mas de seguridad a comparación de Windows pero no es tan personalizable

▼ (c) ¿Qué es GNU?

Historia y Relación entre GNU y Linux

1. **El Proyecto GNU**:

- En 1983, Richard Stallman lanzó el proyecto GNU con el objetivo de crear un sistema operativo libre y completo, similar a Unix. Este sistema operativo debía incluir todas las herramientas necesarias (compiladores, bibliotecas, editores, etc.) que los usuarios y desarrolladores necesitarían para utilizar y desarrollar software.
- El proyecto GNU logró desarrollar la mayoría de los componentes necesarios para un sistema operativo: herramientas como el compilador GCC, la biblioteca estándar C (glibc), y muchas utilidades y aplicaciones.
- **GNU Hurd**: El núcleo del sistema operativo GNU, llamado **GNU Hurd**, comenzó a desarrollarse, pero su desarrollo fue lento y no llegó a un estado de madurez y estabilidad rápidamente.

2. **El Núcleo Linux**:
    - En 1991, **Linus Torvalds**, un estudiante universitario en Finlandia, desarrolló y lanzó el núcleo Linux como un proyecto independiente. Linux es solo el **núcleo** de un sistema operativo (la parte central que gestiona los recursos del hardware y permite que el software se ejecute).
    - El núcleo Linux era un componente que el proyecto GNU aún no había terminado de desarrollar con GNU Hurd. Linux resultó ser estable y rápidamente ganó popularidad entre los desarrolladores.
3. **Unión de GNU y Linux**:
    - La comunidad del software libre, incluyendo a los desarrolladores del proyecto GNU, vio una oportunidad en el núcleo Linux para completar su sistema operativo libre.
    - Al combinar el núcleo Linux con los componentes del sistema GNU (las herramientas y utilidades desarrolladas por el proyecto GNU), nació lo que conocemos como **GNU/Linux**. Este sistema operativo combinó el núcleo Linux con las herramientas y componentes de GNU para crear un sistema operativo completo.
    - Por esta razón, muchos prefieren llamar al sistema operativo completo “GNU/Linux”, reconociendo así la contribución tanto del proyecto GNU

(que proporciona la mayoría de las herramientas y utilidades) como del núcleo Linux.

¿Por qué se le llama "GNU/Linux"?

- **Linux**: Es solo el núcleo del sistema operativo. Sin embargo, por sí solo, no es suficiente para proporcionar un entorno completo de usuario.
- **GNU/Linux**: Es un sistema operativo completo que combina el núcleo Linux con las herramientas y utilidades del proyecto GNU, permitiendo a los usuarios ejecutar aplicaciones y realizar tareas como en cualquier otro sistema operativo

En resumen,
GNU y **Linux** se unieron porque GNU había desarrollado casi todas las partes de un sistema operativo excepto un núcleo funcional, mientras que Linux era un núcleo funcional pero no tenía las herramientas necesarias para ser un sistema operativo completo por sí mismo. Al combinar el núcleo Linux con los componentes GNU, se creó un sistema operativo completo y funcional, que es lo que muchas personas usan hoy en día y que se conoce comúnmente como "Linux", aunque técnicamente se debería llamar "GNU/Linux".

▼ (d) Indique una breve historia sobre la evolución del proyecto GNU

- **El Proyecto GNU**:
  - En 1983, Richard Stallman lanzó el proyecto GNU con el objetivo de crear un sistema operativo libre y completo, similar a Unix. Este sistema operativo debía incluir todas las herramientas necesarias (compiladores, bibliotecas, editores, etc.) que los usuarios y desarrolladores necesitarían para utilizar y desarrollar software.
  - El proyecto GNU logró desarrollar la mayoría de los componentes necesarios para un sistema operativo: herramientas como el

compilador GCC, la biblioteca estándar C (glibc), y muchas utilidades y aplicaciones.

    - **GNU Hurd**: El núcleo del sistema operativo GNU, llamado **GNU Hurd**, comenzó a desarrollarse, pero su desarrollo fue lento y no llegó a un estado de madurez y estabilidad rápidamente.

- **El Núcleo Linux**:
    - En 1991, **Linus Torvalds**, un estudiante universitario en Finlandia, desarrolló y lanzó el núcleo Linux como un proyecto independiente. Linux es solo el **núcleo** de un sistema operativo (la parte central que gestiona los recursos del hardware y permite que el software se ejecute).
    - El núcleo Linux era un componente que el proyecto GNU aún no había terminado de desarrollar con GNU Hurd. Linux resultó ser estable y rápidamente ganó popularidad entre los desarrolladores.
- **Unión de GNU y Linux**:
    - La comunidad del software libre, incluyendo a los desarrolladores del proyecto GNU, vio una oportunidad en el núcleo Linux para completar su sistema operativo libre.
    - Al combinar el núcleo Linux con los componentes del sistema GNU (las herramientas y utilidades desarrolladas por el proyecto GNU), nació lo que conocemos como **GNU/Linux**. Este sistema operativo combinó el núcleo Linux con las herramientas y componentes de GNU para crear un sistema operativo completo.
    - Por esta razón, muchos prefieren llamar al sistema operativo completo “GNU/Linux”, reconociendo así la contribución tanto del proyecto GNU (que proporciona la mayoría de las herramientas y utilidades) como del núcleo 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?

▼ (e) Explique qué es la multitarea, e indique si GNU/Linux hace uso de ella.

multitarea es la capacidad de un sistema operativo para ejecutar múltiples procesos al mismo tiempo, compartiendo recursos como la CPU y la memoria. **GNU/Linux** utiliza la multitarea preventiva para gestionar múltiples procesos de manera eficiente, garantizando que todos los procesos tengan la oportunidad de ejecutarse y manteniendo la estabilidad y capacidad de respuesta del sistema.

Tipos de Multitarea

1. **Multitarea Cooperativa**:
    - En la multitarea cooperativa, cada proceso tiene el control de la CPU hasta que voluntariamente lo cede a otro proceso. Esto significa que si un proceso no cede el control, puede bloquear el sistema o hacer que otras aplicaciones no respondan.
    - Este tipo de multitarea era común en sistemas operativos más antiguos, como las primeras versiones de Windows (antes de Windows 95).
2. **Multitarea Preventiva**:
    - En la multitarea preventiva, el sistema operativo controla la asignación de tiempo de CPU a cada proceso, lo que garantiza que todos los

procesos obtengan una oportunidad para ejecutarse.

- El sistema operativo utiliza temporizadores y otras técnicas para forzar que un proceso ceda el control de la CPU, permitiendo que otros procesos se ejecuten. Esto mejora la estabilidad y la capacidad de respuesta del sistema.
- La mayoría de los sistemas operativos modernos, incluidos GNU/Linux, Windows, y macOS, utilizan multitarea preventiva.

▼ ¿Qué es POSIX?

POSIX (Portable Operating System Interface) es un conjunto de estándares definidos por la IEEE (Instituto de Ingenieros Eléctricos y Electrónicos) que especifica las interfaces de programación de aplicaciones (APIs), junto con las interfaces de comandos y herramientas de software que deben ser compatibles para mantener la portabilidad de aplicaciones entre sistemas operativos que se asemejan a Unix.

Objetivos y Propósito de POSIX

1. **Portabilidad**: El principal objetivo de POSIX es facilitar la portabilidad de aplicaciones a través de diferentes sistemas operativos Unix-like. Al seguir las especificaciones de POSIX, un programa escrito para un sistema operativo compatible con POSIX debería compilarse y ejecutarse en cualquier otro sistema compatible con POSIX con pocas o ninguna modificación.
2. **Interoperabilidad**: POSIX también busca asegurar que los sistemas operativos proporcionen un conjunto común de comandos y utilidades de shell, así como un entorno de desarrollo de software consistente, de modo que los desarrolladores puedan escribir y ejecutar scripts y herramientas de manera uniforme en diferentes plataformas.
3. **Estándares de Programación**: POSIX define estándares para programación, incluyendo la manera en que los programas interactúan con el sistema operativo para realizar tareas comunes como manejo de archivos, manejo de procesos, señales, hilos, sincronización, entrada/salida, entre otros.

2:

Distribuciones de GNU/Linux:

▼ ¿Qué es una distribución de GNU/Linux? Nombre al menos 4 distribuciones de GNU/
Linux y cite diferencias básicas entre ellas.

Una **distribución de GNU/Linux** (o simplemente distribución de Linux) es una versión del sistema operativo Linux que incluye el núcleo Linux junto con un conjunto de software adicional, herramientas y utilidades para formar un sistema operativo completo. Las distribuciones de Linux pueden variar en términos de su enfoque, facilidad de uso, administración de paquetes y objetivos específicos.

Características Comunes de las Distribuciones de GNU/Linux

1. **Núcleo Linux**: Todas las distribuciones utilizan el núcleo Linux, que es el corazón del sistema operativo.
2. **Gestor de Paquetes**: Cada distribución tiene su propio sistema de gestión de paquetes para instalar, actualizar y eliminar software. Ejemplos incluyen `apt` en Debian/Ubuntu, `yum`/`dnf` en Fedora, y `pacman` en Arch Linux.
3. **Entorno de Escritorio**: Las distribuciones pueden incluir diferentes entornos de escritorio, como GNOME, KDE Plasma, Xfce, entre otros, que proporcionan la interfaz gráfica de usuario (GUI).
4. **Software Adicional**: Además del núcleo, cada distribución incluye una selección de software y herramientas preinstaladas, como navegadores web, editores de texto, y aplicaciones de oficina.

- **Ubuntu**
  - **Enfoque**: Orientado a la facilidad de uso y la accesibilidad para usuarios nuevos y para ambientes de escritorio.
  - **Gestor de Paquetes**: `apt` (Advanced Package Tool), utiliza paquetes `.deb`.

    - **Entorno de Escritorio Predeterminado**: GNOME (anteriormente utilizaba Unity).
    - **Ciclo de Lanzamiento**: Ubuntu tiene un ciclo de lanzamiento regular con versiones LTS (Long Term Support) cada dos años, que reciben soporte durante 5 años.
    - **Usos Típicos**: Escritorios, servidores, y entornos de nube.
- **Fedora**
    - **Enfoque**: Focalizado en ofrecer software de vanguardia y nuevas tecnologías, con un ciclo de lanzamiento rápido.
    - **Gestor de Paquetes**: `dnf` (Dandified YUM), utiliza paquetes `.rpm`.
    - **Entorno de Escritorio Predeterminado**: GNOME.
    - **Ciclo de Lanzamiento**: Fedora sigue un ciclo de lanzamiento de aproximadamente cada seis meses, proporcionando software actualizado y características nuevas.
    - **Usos Típicos**: Escritorios, servidores, y desarrollo de software.
- **Debian**
    - **Enfoque**: Conocido por su estabilidad y robustez. Utilizado como base para muchas otras distribuciones, incluida Ubuntu.
    - **Gestor de Paquetes**: `apt`, utiliza paquetes `.deb`.
    - **Entorno de Escritorio Predeterminado**: No tiene un entorno de escritorio predeterminado; los usuarios pueden elegir entre GNOME, KDE Plasma, Xfce, entre otros durante la instalación.
    - **Ciclo de Lanzamiento**: Debian tiene un ciclo de lanzamiento menos predecible, enfocándose en liberar versiones estables cuando el equipo considera que están listas.
    - **Usos Típicos**: Servidores, estaciones de trabajo, y sistemas de misión crítica.
- **Arch Linux**

- **Enfoque**: Enfocado en la simplicidad y el control total del usuario, proporcionando una experiencia de "hazlo tú mismo" (DIY) con una instalación mínima.
- **Gestor de Paquetes**: `pacman`, utiliza paquetes `.pkg.tar.xz`.
- **Entorno de Escritorio Predeterminado**: Ninguno, el usuario debe instalar y configurar su propio entorno de escritorio y software.
- **Ciclo de Lanzamiento**: Rolling release, lo que significa que se actualiza continuamente en lugar de tener versiones estables fijas.
- **Usos Típicos**: Usuarios avanzados y entusiastas que desean un control total sobre su sistema y prefieren configurar cada aspecto del mismo.

▼ (b) ¿En qué se diferencia una distribución de otra?

Las **distribuciones de GNU/Linux** pueden diferenciarse en varios aspectos, desde el núcleo del sistema operativo hasta la selección de software y la filosofía de diseño. Aquí se destacan las principales diferencias entre las distribuciones:

1. Núcleo

- **Núcleo**: Aunque todas las distribuciones usan el núcleo Linux, pueden usar diferentes versiones del núcleo o parches específicos para adaptar el sistema a necesidades particulares.

2. Gestor de Paquetes

- **Gestores de Paquetes**: Cada distribución tiene su propio sistema de gestión de paquetes, que facilita la instalación, actualización y eliminación de software. Ejemplos incluyen:
  - **Debian/Ubuntu**: `apt` (Advanced Package Tool) con paquetes `.deb`.
  - **Fedora/RHEL/CentOS**: `dnf` (Dandified YUM) con paquetes `.rpm`.
  - **Arch Linux**: `pacman` con paquetes `.pkg.tar.xz`.

3. Entorno de Escritorio

- **Entornos de Escritorio**: Las distribuciones pueden ofrecer diferentes entornos de escritorio, que afectan la apariencia y la experiencia del usuario. Ejemplos incluyen:
  - **GNOME**: Utilizado por Debian, Fedora, y Ubuntu.
  - **KDE Plasma**: Utilizado por KDE Neon, Kubuntu, y openSUSE.
  - **Xfce**: Utilizado por Xubuntu y Linux Mint XFCE.

4. Ciclo de Lanzamiento

- **Ciclo de Lanzamiento**: Las distribuciones pueden tener diferentes ciclos de lanzamiento:
  - **Lanzamiento Regular**: Actualizaciones regulares y versiones nuevas cada pocos meses (ej., Fedora, Ubuntu).
  - **Rolling Release**: Actualizaciones continuas sin versiones específicas (ej., Arch Linux, openSUSE Tumbleweed).
  - **LTS (Long Term Support)**: Versiones con soporte prolongado (ej., Ubuntu LTS, Debian Stable).

5. Filosofía y Enfoque

- **Filosofía y Objetivos**: Las distribuciones pueden tener diferentes enfoques y objetivos:
  - **Facilidad de Uso**: Distribuciones como **Ubuntu** y **Linux Mint** se enfocan en ser accesibles para usuarios nuevos con configuraciones listas para usar y una amplia gama de software preinstalado.
  - **Personalización y Control**: Distribuciones como **Arch Linux** ofrecen una instalación mínima para que los usuarios configuren el sistema a su gusto.
  - **Estabilidad y Seguridad**: Distribuciones como **Debian** y **RHEL/CentOS** priorizan la estabilidad y el soporte a largo plazo, siendo comunes en entornos de servidor.

6. Soporte y Comunidad

- **Soporte y Comunidad**: Las distribuciones pueden diferir en el nivel de soporte proporcionado y el tamaño de la comunidad:
  - **Distribuciones Comerciales**: Como **Red Hat Enterprise Linux (RHEL)** y **SUSE Linux Enterprise Server (SLES)**, ofrecen soporte técnico profesional.
  - **Distribuciones Comunitarias**: Como **Debian** y **Arch Linux**, dependen de la comunidad para el soporte y la contribución.

7. Configuración y Herramientas

- **Herramientas de Configuración**: Diferentes distribuciones pueden incluir distintas herramientas para la configuración del sistema:
  - **Ubuntu**: Utiliza `Ubuntu Software` para la gestión de software y `GNOME Control Center` para la configuración del sistema.
  - **Arch Linux**: Utiliza herramientas como `pacman` y `archlinux-config` para la administración del sistema.

Ejemplos de Distribuciones y sus Características

- **Debian**:
  - **Enfoque**: Estabilidad y libertad de software.
  - **Gestor de Paquetes**: `apt`.
  - **Entorno de Escritorio**: Variado (GNOME, KDE, Xfce, etc.).
- **Fedora**:
  - **Enfoque**: Innovación y nuevas tecnologías.
  - **Gestor de Paquetes**: `dnf`.
  - **Entorno de Escritorio**: GNOME por defecto.
- **Arch Linux**:
  - **Enfoque**: Minimalismo y control total del usuario.
  - **Gestor de Paquetes**: `pacman`.

    - **Entorno de Escritorio**: Ninguno por defecto, el usuario elige y configura.
- **Ubuntu**:
    - **Enfoque**: Facilidad de uso y soporte a largo plazo.
    - **Gestor de Paquetes**: `apt`.
    - **Entorno de Escritorio**: GNOME (anteriormente Unity).

Resumen

Las diferencias entre distribuciones de GNU/Linux abarcan el núcleo del sistema, el gestor de paquetes, el entorno de escritorio, el ciclo de lanzamiento, la filosofía, el soporte y las herramientas de configuración. Estas diferencias permiten que cada distribución se adapte a necesidades específicas y preferencias de los usuarios, desde principiantes hasta expertos.

▼ Qué es Debian? Acceda al sitio 1 e indique cuáles son los objetivos del proyecto y
una breve cronología del mismo

Debian es una de las distribuciones de GNU/Linux más antiguas y respetadas. Es conocida por su estabilidad, su enfoque en el software libre, y su robustez. Aquí tienes una descripción detallada de Debian, incluyendo sus objetivos y una breve cronología:

¿Qué es Debian?

Debian es una distribución de GNU/Linux que utiliza el núcleo Linux junto con un amplio conjunto de software libre. Está desarrollada y mantenida por una comunidad global de voluntarios y se caracteriza por su estabilidad y su compromiso con el software libre.

Objetivos del Proyecto Debian

1. **Estabilidad y Confiabilidad**:

- **Objetivo**: Proporcionar un sistema operativo estable y confiable, adecuado para servidores, estaciones de trabajo y entornos de misión crítica.
- **Implementación**: Se logra a través de rigurosas pruebas y la integración de software maduro y bien probado.

2. **Software Libre**:
    - **Objetivo**: Promover el uso del software libre y garantizar que la mayoría del software incluido sea de código abierto.
    - **Implementación**: Debian sigue estrictamente las Directrices de Software Libre de Debian (DFSG), que definen qué software puede incluirse en la distribución.
3. **Universalidad y Flexibilidad**:
    - **Objetivo**: Ser una distribución universal que pueda adaptarse a diferentes arquitecturas de hardware y necesidades de usuarios.
    - **Implementación**: Debian ofrece soporte para una amplia gama de arquitecturas de hardware y permite una gran personalización y configuración.
4. **Gestión Comunitaria**:
    - **Objetivo**: Operar como un proyecto de código abierto basado en la colaboración y el consenso de su comunidad de desarrolladores y usuarios.
    - **Implementación**: Las decisiones se toman de manera democrática a través de votaciones entre los miembros del Proyecto Debian.
5. **Ciclo de Lanzamiento Sostenible**:
    - **Objetivo**: Mantener un ciclo de lanzamiento que proporcione versiones estables y seguras con actualizaciones regulares.
    - **Implementación**: Debian tiene un ciclo de lanzamiento que no está basado en fechas específicas, sino en la madurez y estabilidad de las versiones.

Cronología Breve de Debian

1. **1993**:
    - **Inicio del Proyecto**: Ian Murdock anuncia el proyecto Debian, nombrado en honor a su novia Debra y a él mismo. El objetivo inicial es crear una distribución de Linux que sea completamente libre y compatible con las ideas del software libre.
2. **1994**:
    - **Primera Versión**: Se lanza la primera versión oficial de Debian, la versión 0.91, que ya incluye una colección de paquetes y el núcleo Linux.
3. **1996**:
    - **Primera Versión Estable**: Se lanza Debian 1.1, conocida como "Buzz". Esta es la primera versión estable del sistema y marca el inicio de las versiones oficiales de Debian.
4. **1999**:
    - **Debian 2.0 "Hamm"**: Introducción de una nueva infraestructura de paquetes y soporte para más arquitecturas. Debian se consolida como una distribución estable y de alta calidad.
5. **2001**:
    - **Debian 3.0 "Woody"**: Esta versión es notable por su enfoque en la estabilidad y la mejora de la gestión de paquetes, y por ofrecer soporte para una amplia gama de hardware.
6. **2004**:
    - **Debian 3.1 "Sarge"**: Introducción de mejoras en la instalación y soporte para más plataformas. Este lanzamiento también trae una nueva política de seguridad y soporte a largo plazo.
7. **2006**:
    - **Debian 4.0 "Etch"**: Incluye mejoras en la instalación y la administración del sistema, con un enfoque en la facilidad de uso y la estabilidad.

8. **2010**:
    - **Debian 6.0 "Squeeze"**: Introduce soporte para la arquitectura ARM y varias mejoras en el sistema de paquetes y la instalación.
9. **2013**:
    - **Debian 7.0 "Wheezy"**: Incluye mejoras en el instalador y actualizaciones significativas en software y seguridad.
10. **2015**:
    - **Debian 8.0 "Jessie"**: Marca la introducción del sistema de inicio `systemd` como opción predeterminada, junto con varias mejoras en la seguridad y el rendimiento.
11. **2017**:
    - **Debian 9.0 "Stretch"**: Introduce mejoras en la seguridad y el soporte de hardware, así como nuevas versiones de software.
12. **2019**:
    - **Debian 10.0 "Buster"**: Enfocado en la estabilidad y la seguridad, con nuevas versiones de paquetes y mejoras en la instalación y el soporte de hardware.
13. **2021**:
    - **Debian 11.0 "Bullseye"**: Incluye actualizaciones en el núcleo, el software y la infraestructura del sistema, continuando con el enfoque en la estabilidad y el soporte a largo plazo.

Resumen

Debian es una distribución de GNU/Linux conocida por su estabilidad, compromiso con el software libre y enfoque comunitario. Desde su inicio en 1993, ha evolucionado para convertirse en una de las distribuciones más importantes y respetadas en el ecosistema de GNU/Linux. Sus objetivos incluyen proporcionar un sistema operativo universal, estable y libre, mientras se mantiene fiel a los principios del software libre.

3 Estructura de GNU/Linux:

▼ Nombre cuales son los 3 componentes fundamentales de GNU/Linux.

(a) Componentes Fundamentales de GNU/Linux

1. **Núcleo (Kernel)**:
   - **Descripción**: El núcleo es la parte central del sistema operativo. En el caso de GNU/Linux, el núcleo es **Linux**, desarrollado por Linus Torvalds y mantenido por la comunidad. El núcleo gestiona el hardware del sistema, la memoria, los procesos, y las comunicaciones entre el hardware y el software.
   - **Funciones**: Controla el acceso al hardware, maneja la memoria, administra los procesos, y proporciona servicios esenciales para el sistema operativo y las aplicaciones.
2. **Sistema GNU**:
   - **Descripción**: El sistema GNU proporciona las herramientas y utilidades básicas que forman el entorno de usuario del sistema operativo. Estas herramientas incluyen el compilador GCC, la biblioteca estándar C (glibc), el shell Bash, y otras utilidades esenciales.
   - **Funciones**: Ofrece herramientas para el manejo de archivos, la ejecución de comandos, la programación, y otras tareas básicas necesarias para el funcionamiento del sistema operativo.
3. **Sistema de Gestión de Paquetes**:
   - **Descripción**: El sistema de gestión de paquetes permite la instalación, actualización y eliminación de software en el sistema. Cada distribución de GNU/Linux utiliza su propio sistema de gestión de paquetes.
   - **Ejemplos**:
     - **Debian/Ubuntu**: `apt` (Advanced Package Tool) con paquetes `.deb`.

- **Fedora/RHEL/CentOS**: `dnf` (Dandified YUM) con paquetes `.rpm`.
- **Arch Linux**: `pacman` con paquetes `.pkg.tar.xz`.

▼ Mencione y explique la estructura básica del Sistema Operativo GNU/Linux.

La estructura de un sistema GNU/Linux está organizada en varias capas y directorios que facilitan la administración y el uso del sistema. Aquí están los componentes clave:

1. **Núcleo (Kernel)**:
   - **Ubicación**: Generalmente se encuentra en `/boot` como un archivo llamado `vmlinuz` o similar.
   - **Función**: Controla el hardware y proporciona servicios esenciales para el funcionamiento del sistema operativo.
2. **Sistema de Archivos**:
   - **Raíz (`/`)**: El punto de partida de todo el sistema de archivos. Todos los archivos y directorios están organizados bajo este directorio.
   - **Directorios Principales**:
     - `/bin`: Contiene los comandos básicos del sistema, como `ls`, `cp`, `mv`, y otros programas esenciales para el funcionamiento básico del sistema.
     - `/boot`: Contiene los archivos necesarios para el arranque del sistema, incluidos el núcleo Linux y los archivos de configuración del cargador de arranque.
     - `/etc`: Contiene archivos de configuración del sistema y de las aplicaciones. Ejemplos son `fstab` (configuración de sistemas de archivos) y `passwd` (usuarios del sistema).
     - `/home`: Directorio donde se almacenan los archivos personales de los usuarios. Cada usuario tiene su propio subdirectorio en `/home`.

- `/lib`: Contiene bibliotecas compartidas esenciales para los programas en `/bin` y `/sbin`.
- `/media`: Punto de montaje para dispositivos de almacenamiento extraíbles como CD, DVD y unidades USB.
- `/mnt`: Punto de montaje temporal para sistemas de archivos, a menudo usado para montar manualmente dispositivos de almacenamiento.
- `/opt`: Contiene paquetes de software opcionales o de terceros que no forman parte del sistema base.
- `/proc`: Sistema de archivos virtual que proporciona información sobre el estado del núcleo y los procesos en ejecución.
- `/root`: Directorio personal del superusuario (root). No debe confundirse con `/home/root`.
- `/sbin`: Contiene comandos esenciales para la administración del sistema, accesibles principalmente para el superusuario (root).
- `/srv`: Contiene datos para servicios proporcionados por el sistema, como sitios web o bases de datos.
- `/tmp`: Directorio para archivos temporales que pueden ser eliminados al reiniciar el sistema.
- `/usr`: Contiene aplicaciones y archivos de uso compartido, como programas y bibliotecas que no son esenciales para el arranque del sistema, pero que son necesarios para la operación normal.
- `/var`: Contiene archivos variables que cambian con el tiempo, como archivos de registro (`logs`) y cachés de aplicaciones.

3. **Sistema de Gestión de Paquetes**:
   - **Descripción**: Herramientas y bases de datos que permiten la instalación y administración del software en el sistema.
   - **Ejemplos**:

- `apt` (Debian/Ubuntu): Herramienta de línea de comandos para la gestión de paquetes.
- `rpm` (Red Hat/CentOS): Formato de paquete y herramienta de gestión.
- `pacman` (Arch Linux): Herramienta de gestión de paquetes.

4. **Interfaz de Usuario**:
   - **Entorno de Escritorio**: Las distribuciones pueden ofrecer diferentes entornos de escritorio, como GNOME, KDE Plasma, Xfce, entre otros.
   - **Shell**: La interfaz de línea de comandos proporcionada por el shell, como `bash` o `zsh`, que permite a los usuarios ejecutar comandos y scripts.

Resumen

La estructura básica de un sistema GNU/Linux se basa en la combinación del núcleo Linux, las herramientas del sistema GNU, y un sistema de archivos organizado en una jerarquía que facilita el acceso y la administración del sistema. La gestión de paquetes permite la instalación y actualización de software, mientras que la interfaz de usuario puede variar según las preferencias de los usuarios y las características de la distribución.

4 Kernel:

▼ ¿Qué es? Indique una breve reseña histórica acerca de la evolución del Kernel de
GNU/Linux.

El kernel de Linux es el núcleo del sistema operativo Linux, que es el componente central que interactúa directamente con el hardware del sistema y gestiona los recursos del ordenador. A continuación, se presenta una breve reseña histórica de su evolución:

1. **Creación y lanzamiento inicial (1991):** El kernel de Linux fue creado por Linus Torvalds, un estudiante de informática finlandés, en 1991. Linus buscaba un sistema operativo libre similar a Unix que pudiera

usar en su computadora personal. Publicó la primera versión del kernel, la 0.01, en septiembre de 1991 y la distribuyó bajo la licencia GPL (Licencia Pública General de GNU) en 1992.

2. **Rápido desarrollo y adopción (1992-1996):** Tras su lanzamiento, Linux atrajo rápidamente a una comunidad de desarrolladores y entusiastas que contribuyeron al desarrollo del kernel y la creación de software adicional para crear un sistema operativo completo. Durante este período, Linux comenzó a ser utilizado en servidores y entornos de investigación.
3. **Ampliación de soporte y características (1997-2005):** Linux evolucionó rápidamente, añadiendo soporte para más arquitecturas de hardware, sistemas de archivos y características avanzadas como la multitarea, la gestión de memoria mejorada y la seguridad. Las distribuciones de Linux, como Debian, Red Hat y SUSE, se volvieron populares, ofreciendo versiones completas de sistemas operativos basados en el kernel de Linux.
4. **Consolidación y expansión (2006-2014):** Durante estos años, el kernel de Linux se consolidó como una opción robusta para servidores y sistemas integrados. Las grandes empresas tecnológicas comenzaron a adoptarlo, y surgieron distribuciones como Ubuntu, que facilitaron el uso de Linux en escritorios personales. Además, el kernel comenzó a ser utilizado en una amplia gama de dispositivos, desde teléfonos inteligentes (con Android) hasta supercomputadoras.
5. **Avances y modernización (2015-presente):** En los últimos años, el kernel de Linux ha seguido evolucionando, mejorando en áreas como la seguridad, el rendimiento, la virtualización y el soporte de hardware moderno. Con la creciente importancia de la nube, Linux se ha convertido en el núcleo de muchas infraestructuras en la nube, con proveedores como Google, Amazon y Microsoft que utilizan Linux ampliamente en sus servicios.

El kernel de Linux ha evolucionado desde un proyecto personal de un estudiante a ser el núcleo de uno de los sistemas operativos más importantes del mundo, utilizado en una amplia variedad de aplicaciones

desde servidores empresariales hasta dispositivos móviles y sistemas embebidos.

▼ ¿Cuáles son sus funciones principales?

El kernel de Linux, como núcleo del sistema operativo, tiene varias funciones clave que son esenciales para el funcionamiento del sistema. Sus principales funciones son:

1. **Gestión de procesos:** El kernel gestiona la creación, ejecución y terminación de procesos. También controla la asignación de tiempo de CPU para cada proceso, asegurando que múltiples procesos puedan ejecutarse simultáneamente (multitarea) sin interferir entre sí.
2. **Gestión de memoria:** Controla la asignación y liberación de memoria RAM para los procesos en ejecución, asegurando que cada proceso tenga acceso a la memoria que necesita sin interferir con otros procesos. Esto incluye la gestión de la memoria virtual, que permite que los procesos utilicen más memoria de la disponible físicamente mediante el uso de un espacio de intercambio en el disco.
3. **Gestión de dispositivos:** El kernel actúa como un intermediario entre el hardware del sistema (como discos duros, tarjetas gráficas, teclados, etc.) y el software. Utiliza controladores de dispositivos para interactuar con el hardware, proporcionando una interfaz uniforme para que los procesos puedan acceder a los dispositivos sin tener que preocuparse por los detalles específicos del hardware.
4. **Gestión de sistemas de archivos:** Maneja la lectura y escritura de datos en los sistemas de archivos. El kernel proporciona acceso a los sistemas de archivos, permitiendo a los usuarios y aplicaciones almacenar y recuperar datos de manera organizada en los discos duros y otros dispositivos de almacenamiento.
5. **Gestión de seguridad y permisos:** Implementa mecanismos de seguridad que controlan el acceso a los recursos del sistema. Esto incluye la gestión de permisos de archivos y la separación de procesos

para evitar que los procesos maliciosos o defectuosos comprometan el sistema.

6. **Gestión de redes:** Proporciona una pila de red que permite la comunicación entre dispositivos a través de redes locales o internet. Esto incluye el manejo de protocolos de red como TCP/IP, y la gestión de interfaces de red, enrutamiento de paquetes y conexiones de red.
7. **Control de sistemas de energía:** Maneja las funciones de ahorro de energía y administración de energía del sistema, como el apagado de dispositivos no utilizados y la implementación de diferentes estados de energía para el sistema.
8. **Intercomunicación entre procesos (IPC):** Proporciona mecanismos que permiten a los procesos comunicarse entre sí y sincronizar sus acciones. Esto es esencial para la cooperación entre aplicaciones y para compartir recursos.

Estas funciones permiten que el kernel de Linux actúe como el componente central que facilita la interacción entre el software y el hardware del sistema, proporcionando un entorno estable y eficiente para la ejecución de aplicaciones y servicios.

▼ ¿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?

La versión actual del kernel Linux es la 6.10, lanzada recientemente con varias mejoras y nuevas características, incluyendo soporte mejorado para hardware, mejoras de seguridad, y nuevas funcionalidades para desarrolladores y usuarios(

OMG! Ubuntu

).

Esquema de versionado del kernel Linux

Versionado antes de la versión 2.4: Antes de la versión 2.4, Linux utilizaba un esquema de versionado donde el número de versión consistía en tres partes: *X.Y.Z*. En este esquema, el segundo número (*Y*) era el más relevante, ya que indicaba si una versión era estable o de desarrollo. Los números impares (por ejemplo, 2.1, 2.3) indicaban versiones de desarrollo, mientras que los números pares (por ejemplo, 2.2, 2.4) correspondían a versiones estables. Esto permitía a los desarrolladores trabajar en nuevas características y experimentos en las versiones de desarrollo sin comprometer la estabilidad de las versiones principales estables.

Cambios a partir de la versión 2.6: A partir de la versión 2.6, el esquema de versionado cambió. Linux adoptó un enfoque donde todos los lanzamientos eran considerados como versiones estables, eliminando la distinción entre números de versión pares e impares para desarrollo y estabilidad. Desde entonces, las versiones se numeran secuencialmente, independientemente del tipo de cambios introducidos, y las versiones de prelanzamiento o de prueba se indican con el sufijo "-rc" (release candidate). Este cambio simplificó el desarrollo y la adopción de nuevas versiones del kernel(

Kernel.org

).

▼ ¿Es posible tener más de un Kernel de GNU/Linux instalado en la misma máquina?

Sí, es posible tener más de un kernel de GNU/Linux instalado en la misma máquina. De hecho, es una práctica común en los sistemas Linux, especialmente en entornos de desarrollo o pruebas, donde se necesita probar nuevas versiones del kernel o mantener una versión anterior como respaldo.

Cómo funciona tener múltiples kernels instalados

1. **Instalación de múltiples kernels:** Los kernels en Linux están instalados en directorios separados en el sistema de archivos, típicamente en `/boot`. Cada kernel tiene su propio archivo de imagen (`vmlinuz` o similar)

y puede tener módulos asociados que se almacenan en subdirectorios específicos bajo `/lib/modules`.

2. **Gestor de arranque:** Al tener múltiples kernels instalados, el gestor de arranque (como GRUB o LILO) se configura para reconocer y listar cada kernel disponible. Cuando la computadora se inicia, el usuario puede seleccionar el kernel que desea arrancar desde el menú del gestor de arranque. Esto permite cambiar entre diferentes versiones del kernel sin necesidad de desinstalarlas.
3. **Mantenimiento de sistemas estables:** Tener múltiples kernels permite a los usuarios probar nuevas versiones del kernel sin comprometer la estabilidad del sistema. Si un nuevo kernel causa problemas, el usuario puede simplemente reiniciar y seleccionar un kernel anterior desde el menú del gestor de arranque.
4. **Actualizaciones y pruebas:** Al actualizar el kernel, es común mantener la versión anterior hasta asegurarse de que el nuevo kernel funciona correctamente. Esto es especialmente importante en servidores y entornos críticos, donde la estabilidad es esencial.

Ejemplo práctico

Supongamos que un usuario tiene la versión del kernel 5.15 instalada y decide instalar el kernel 6.1 para aprovechar nuevas características. El nuevo kernel se instalará junto al antiguo, y el gestor de arranque se actualizará para incluir ambas versiones. Durante el inicio, el usuario verá ambas opciones y podrá seleccionar cuál cargar.

En resumen, tener múltiples kernels instalados proporciona flexibilidad y seguridad, permitiendo que los sistemas Linux sean más robustos y adaptables a diferentes necesidades.

▼ (e) ¿Dónde se encuentra ubicado dentro del File System?

En un sistema operativo Linux, los kernels instalados se encuentran ubicados principalmente en el directorio `/boot` del sistema de archivos.

Ubicación del Kernel dentro del Sistema de Archivos

1. **Directorio `/boot`:**
    - El directorio `/boot` es donde se almacenan los archivos necesarios para que el sistema operativo Linux arranque. Esto incluye los archivos del kernel, el inicializador de RAM (`initrd` o `initramfs`), el cargador de arranque (como GRUB), y otros archivos de configuración relacionados.
    - Los archivos del kernel suelen estar nombrados como `vmlinuz-x.y.z`, donde `x.y.z` indica la versión del kernel. Por ejemplo, un archivo llamado `vmlinuz-5.15.0` sería el kernel de la versión 5.15.0 de Linux.
    - También puede haber imágenes comprimidas del kernel (como `vmlinuz-`) y módulos del kernel en `/lib/modules`.
2. **Archivos de configuración del cargador de arranque:**
    - Dentro de `/boot`, también se encuentran archivos de configuración utilizados por el cargador de arranque (GRUB o LILO, por ejemplo) para cargar el kernel seleccionado al iniciar el sistema. Esto incluye archivos como `grub.cfg` (en el caso de GRUB), que especifica los menús de arranque y las opciones del kernel.

Propósito de la Ubicación

- **Acceso en el arranque:** La ubicación en `/boot` es crucial porque este directorio contiene todo lo necesario para arrancar el sistema operativo. Es la primera parte del sistema de archivos que el cargador de arranque necesita acceder después de que la BIOS o UEFI hayan pasado el control.
- **Organización:** Mantener todos los archivos relacionados con el arranque en `/boot` organiza los archivos críticos del sistema y facilita el mantenimiento, como actualizaciones del kernel o del cargador de arranque.
- **Compatibilidad:** Al mantener los kernels y los archivos de inicio separados del resto del sistema de archivos (como en `/root`, `/home`, etc.), se asegura que el sistema pueda arrancar incluso si hay problemas con otras particiones del disco.

En resumen, el kernel de Linux y los archivos relacionados están ubicados en el directorio `/boot` para garantizar un arranque eficiente y organizado del sistema operativo, proporcionando un acceso rápido y centralizado a todos los elementos necesarios para cargar el sistema operativo en memoria.

▼ (f) ¿El Kernel de GNU/Linux es monolítico? Justifique.

Monolitico : linux
Hibrido: maxOs y Windows

Sí, el núcleo (o kernel) de GNU/Linux es considerado monolítico. La arquitectura monolítica de un kernel significa que el núcleo incluye tanto la gestión del hardware como las funciones del sistema operativo en un solo espacio de memoria. Esto contrasta con los kernels microkernel, que mantienen la mayor parte de las funciones del sistema operativo en procesos separados que se comunican a través de mensajes.

En el caso del kernel de GNU/Linux:

1. **Monolítico**: El kernel de GNU/Linux maneja directamente la administración de dispositivos, la gestión de memoria, la gestión de procesos, y otros servicios del sistema operativo. Todos estos servicios están integrados en el mismo espacio de memoria del núcleo.
2. **Modularidad**: Aunque es monolítico, el kernel de GNU/Linux es altamente modular. Esto significa que puede cargar y descargar módulos del núcleo dinámicamente sin necesidad de reiniciar el sistema. Estos módulos pueden incluir drivers de hardware y otros componentes que extienden las capacidades del núcleo sin necesidad de recompilarlo.
3. **Ventajas**: La arquitectura monolítica puede ofrecer un mejor rendimiento en comparación con los microkernels, ya que las llamadas al sistema y la comunicación entre componentes son más rápidas al estar en el mismo espacio de memoria.

4. **Desventajas**: Sin embargo, la integración de todos los servicios en un solo espacio de memoria puede aumentar el riesgo de fallos del sistema en caso de errores en el núcleo.

En resumen, el kernel de GNU/Linux es monolítico, pero combina esta característica con una arquitectura modular que le permite adaptarse a diversas necesidades y configuraciones.

5 Intérprete de comandos (Shell)):

▼ (a) ¿Qué es?

Un intérprete de comandos, o **shell**, es un programa que proporciona una interfaz para que los usuarios interactúen con el sistema operativo. A través de la shell, los usuarios pueden ejecutar comandos, lanzar programas y gestionar el sistema de archivos.

Aquí tienes una descripción más detallada de qué es y qué hace un shell:

1. **Interfaz de Usuario**: El shell actúa como un intermediario entre el usuario y el sistema operativo. Permite a los usuarios introducir comandos en una línea de texto (en una terminal) para que el sistema operativo los ejecute.
2. **Ejecución de Comandos**: Los comandos que se ingresan en el shell pueden ser programas ejecutables, scripts, o comandos internos del shell que realizan tareas específicas, como la gestión de archivos y directorios.
3. **Scripts y Automatización**: Los shells permiten la creación de scripts, que son archivos de texto que contienen una serie de comandos que se ejecutan en secuencia. Esto es útil para automatizar tareas repetitivas o complejas.
4. **Redirección y Pipes**: Los shells permiten redirigir la entrada y salida de comandos y utilizar pipes para conectar la salida de un comando a la entrada de otro. Esto facilita la creación de secuencias de procesamiento de datos.

5. **Variables y Control de Flujo**: Los shells soportan variables y estructuras de control de flujo, como bucles y condicionales, que permiten escribir scripts más avanzados.
6. **Interactividad**: Los shells pueden funcionar en modo interactivo (donde el usuario ingresa comandos manualmente) o en modo no interactivo (donde se ejecutan scripts sin interacción directa).
7. **Tipos de Shells**: Existen varios tipos de shells, cada uno con características y sintaxis específicas. Algunos ejemplos incluyen:
    - **Bash** (Bourne Again Shell): Ampliamente utilizado en sistemas GNU/Linux y macOS.
    - **Zsh** (Z Shell): Ofrece características avanzadas y personalización.
    - **Fish** (Friendly Interactive Shell): Conocido por su simplicidad y características amigables para el usuario.
    - **Cmd.exe**: El intérprete de comandos en sistemas Windows.
    - **PowerShell**: Un shell más avanzado en Windows que combina características de shell con un lenguaje de scripting potente.

En resumen, el shell es una herramienta fundamental en la administración y uso de sistemas operativos basados en texto, facilitando la interacción y automatización de tareas.

▼ (b) ¿Cuáles son sus funciones?

- **Ejecución de Comandos**: Permite a los usuarios ejecutar comandos para realizar tareas como manipulación de archivos, gestión de procesos y configuración del sistema.
- **Interpretación de Scripts**: Ejecuta secuencias de comandos o scripts que automatizan tareas repetitivas.
- **Redirección y Pipes**: Facilita el redireccionamiento de la entrada y salida de comandos y el encadenamiento de comandos mediante pipes para el procesamiento de datos.
- **Interactividad y Control de Flujo**: Permite el uso de estructuras de control (bucles, condicionales) en scripts, así como interactuar con el

usuario mediante la entrada de datos.

- **Gestión de Variables**: Soporta la creación y uso de variables para almacenar datos temporales y configurar el entorno de trabajo.
- **Autocompletado y Ayuda**: Ofrece autocompletado de comandos y nombres de archivos, y proporciona ayuda y documentación sobre comandos y opciones.

▼ (c) Mencione al menos 3 intérpretes de comandos que posee GNU/Linux y compárelos
entre ellos.

- **Bash (Bourne Again Shell)**:
  - **Características**: Ampliamente utilizado, compatible con el shell Bourne original, soporta scripting avanzado, autocompletado, historial de comandos, y un sistema de alias.
  - **Ventajas**: Extensa documentación y soporte, ampliamente disponible en casi todas las distribuciones de GNU/Linux.
  - **Desventajas**: Puede ser más lento en comparación con otros shells más modernos debido a su amplia gama de características.
- **Zsh (Z Shell)**:
  - **Características**: Similar a Bash, pero con características adicionales como una mejor completación de comandos, soporte para temas y plugins (a través de frameworks como Oh My Zsh), y una mayor flexibilidad.
  - **Ventajas**: Potente y altamente personalizable, excelente autocompletado y capacidades de scripting.
  - **Desventajas**: Menos estándar en comparación con Bash, lo que puede requerir configuración adicional en algunos entornos.
- **Fish (Friendly Interactive Shell)**:
  - **Características**: Conocido por su facilidad de uso y características interactivas, como autocompletado inteligente y sugerencias en tiempo real, y una sintaxis amigable.

    - **Ventajas**: Muy fácil de usar para principiantes, configuración automática de autocompletado y un sistema de scripting sencillo.
    - **Desventajas**: Menos compatible con scripts escritos para Bash, lo que puede requerir modificaciones al migrar scripts entre shells.

▼ ¿Dónde se ubican (path) los comandos propios y externos al Shell?

- **Comandos Propios**: Los comandos internos del shell, como `cd`, `echo`, y `pwd`, están integrados en el propio shell y no tienen una ubicación de archivo específica en el sistema.
- **Comandos Externos**: Los comandos externos (programas y utilidades) se encuentran en directorios especificados en la variable de entorno `PATH`. Los directorios comunes incluyen:
    - `/bin`: Contiene comandos esenciales para el funcionamiento básico del sistema.
    - `/usr/bin`: Contiene comandos de usuario y aplicaciones instaladas.
    - `/sbin`: Contiene comandos para la administración del sistema y mantenimiento (generalmente requeridos para el administrador del sistema).

▼ (e) ¿Por qué considera que el Shell no es parte del Kernel de GNU/Linux?

El shell no es parte del kernel de GNU/Linux porque:

- **Separación de Funcionalidades**: El kernel es responsable de la gestión de hardware, la memoria, y la comunicación entre procesos, mientras que el shell proporciona una interfaz de usuario para interactuar con el sistema operativo. Estos son dos niveles de funcionalidad diferentes.
- **Modularidad y Flexibilidad**: El kernel está diseñado para ser modular y eficiente, proporcionando una capa fundamental de servicios. El shell, al ser una interfaz de usuario, está diseñado para ser más flexible y ofrecer funcionalidades adicionales que no forman parte de la gestión básica del sistema.
- **Independencia**: Los shells son programas de usuario que pueden ser reemplazados, actualizados o personalizados sin afectar al núcleo del sistema. Esta separación permite que el sistema operativo se adapte a

diferentes necesidades y preferencias de los usuarios sin modificar el núcleo.

▼ (f) ¿Es posible definir un intérprete de comandos distinto para cada usuario? ¿Desde dónde
se define? ¿Cualquier usuario puede realizar dicha tarea?

Sí, es posible definir un intérprete de comandos distinto para cada usuario. Esto se realiza mediante la configuración del archivo de perfil del usuario. Aquí te explico cómo:

1. **Archivo de Configuración**:
   - Cada usuario puede definir su shell predeterminado en su archivo de configuración, como `.bashrc`, `.zshrc`, o `.config/fish/config.fish`, dependiendo del shell que utilice.
   - Para cambiar el shell predeterminado para un usuario, se puede modificar la entrada correspondiente en el archivo `/etc/passwd`. Por ejemplo, para cambiar el shell de un usuario `username` a `zsh`, el administrador puede editar `/etc/passwd` y cambiar la última columna para `username` a `/bin/zsh`.
2. **Comando `chsh`**:
   - Los usuarios pueden cambiar su shell predeterminado utilizando el comando `chsh` (change shell). Por ejemplo, para cambiar a `zsh`, el usuario puede ejecutar `chsh -s /bin/zsh`.
3. **Permisos**:
   - Los usuarios normales pueden cambiar su propio shell usando `chsh` siempre y cuando el nuevo shell esté en la lista de shells válidos en `/etc/shells`. Solo el administrador (root) puede cambiar el shell de otros usuarios.

En resumen, cualquier usuario puede definir su propio intérprete de comandos si tiene los permisos adecuados y si el nuevo shell está instalado y disponible en el sistema.

6 Sistema de Archivos (File System):

▼ (a) ¿Qué es?

Un **sistema de archivos** es una forma de organizar y gestionar los datos en un dispositivo de almacenamiento, como un disco duro, una SSD o una unidad flash USB. Es responsable de cómo se guardan, acceden y gestionan los datos en el sistema operativo. Un sistema de archivos proporciona una estructura jerárquica que permite almacenar archivos y directorios de manera organizada, facilitando la recuperación, modificación y eliminación de datos.

Funciones principales de un sistema de archivos:

- **Organización de datos**: Proporciona un esquema para almacenar archivos en una estructura jerárquica de directorios.
- **Gestión de espacio**: Administra el espacio disponible en el dispositivo de almacenamiento, asignando bloques para almacenar datos.
- **Acceso y permisos**: Controla quién puede acceder y qué operaciones pueden realizarse en los archivos y directorios.
- **Integridad y seguridad de datos**: Asegura que los datos se mantengan seguros y que la integridad de los mismos se preserve ante errores o fallos.

▼ (b) Mencione sistemas de archivos soportados por GNU/Linux.

GNU/Linux soporta una amplia gama de sistemas de archivos, algunos de los más comunes incluyen:

1. **ext4 (Fourth Extended Filesystem)**: Es el sistema de archivos más utilizado en distribuciones GNU/Linux modernas. Es conocido por su buena performance y características como el soporte de grandes volúmenes y grandes tamaños de archivo, journaling, y soporte extendido de atributos.
2. **ext3 (Third Extended Filesystem)**: Es un sistema de archivos con journaling, similar a ext4 pero con menos características avanzadas. Todavía es usado en sistemas más antiguos.
3. **ext2 (Second Extended Filesystem)**: Un sistema de archivos sin journaling, más simple y utilizado en sistemas más antiguos o en

dispositivos de almacenamiento con limitaciones.

4. **XFS**: Un sistema de archivos de alto rendimiento, conocido por su capacidad de manejar archivos grandes y su escalabilidad en servidores.
5. **Btrfs (B-Tree File System)**: Ofrece características avanzadas como snapshots, compresión y RAID. Diseñado para gestionar grandes volúmenes de datos y facilitar la administración del almacenamiento.
6. **FAT32 (File Allocation Table 32)**: Un sistema de archivos ampliamente compatible, utilizado comúnmente en dispositivos de almacenamiento portátiles como memorias USB. Tiene limitaciones en el tamaño máximo de archivo (4GB) y tamaño máximo de partición (8TB).
7. **exFAT (Extended File Allocation Table)**: Una versión extendida de FAT32, diseñada para superar sus limitaciones. Soporta archivos de mayor tamaño y es común en dispositivos de almacenamiento flash.
8. **NTFS (New Technology File System)**: El sistema de archivos utilizado por Windows. GNU/Linux lo soporta principalmente a través del controlador `ntfs-3g`, que permite lectura y escritura.
9. **ReiserFS**: Un sistema de archivos que fue popular en el pasado, conocido por su buena performance con archivos pequeños.

▼ (c) ¿Es posible visualizar particiones del tipo FAT y NTFS en GNU/Linux?

Sí, es posible visualizar y montar particiones del tipo FAT y NTFS en GNU/Linux.

- **FAT (FAT12, FAT16, FAT32)**: GNU/Linux tiene soporte nativo para sistemas de archivos FAT. Las particiones FAT se pueden montar fácilmente y acceder a ellas usando comandos estándar como `mount`.
- **NTFS**: Para soportar NTFS, GNU/Linux utiliza el controlador `ntfs-3g`, que proporciona capacidades de lectura y escritura para las particiones NTFS. Este controlador suele estar disponible en la mayoría de las distribuciones y permite montar y acceder a particiones NTFS de manera similar a otros sistemas de archivos.

▼ ¿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?

La estructura de los sistemas de archivos en GNU/Linux sigue el estándar **Filesystem Hierarchy Standard (FHS)**, que define una organización jerárquica para los archivos y directorios.

Directorio Raíz (`/`): Es el nivel superior del sistema de archivos. Todos los demás directorios y archivos se encuentran dentro de este directorio.

Directorio raíz y los más importantes en GNU/Linux:

1. **/bin**: Contiene los binarios esenciales del sistema, disponibles tanto para el superusuario como para usuarios normales (por ejemplo, comandos básicos como `ls`, `cp`, `mv`).
2. **/boot**: Contiene archivos necesarios para el arranque del sistema, incluidos el kernel de Linux y los archivos de configuración del gestor de arranque (por ejemplo, GRUB).
3. **/dev**: Contiene archivos de dispositivos especiales que representan el hardware del sistema (por ejemplo, discos duros, unidades USB).
4. **/etc**: Contiene archivos de configuración de todo el sistema, incluyendo scripts de inicio, configuraciones de red, y configuraciones de servicios.
5. **/home**: Directorio que contiene los directorios personales de los usuarios. Cada usuario tiene un subdirectorio (por ejemplo, `/home/usuario`) donde se almacenan sus archivos personales y configuraciones.
6. **/lib**: Contiene bibliotecas esenciales del sistema para los binarios ubicados en `/bin` y `/sbin`.
7. **/media**: Punto de montaje para medios removibles, como CD-ROMs, DVDs y unidades USB.
8. **/mnt**: Directorio utilizado para montar sistemas de archivos temporalmente, como discos duros adicionales.

9. **/opt**: Contiene software y aplicaciones adicionales instalados por el usuario o software de terceros.
10. **/root**: Directorio personal del usuario root (administrador del sistema).
11. **/sbin**: Contiene binarios esenciales para la administración del sistema, disponibles principalmente para el superusuario (por ejemplo, comandos como `fdisk`, `ifconfig`).
12. **/srv**: Contiene datos de servicios proporcionados por el sistema, como archivos de servidores web o FTP.
13. **/tmp**: Directorio para archivos temporales que pueden ser utilizados por diferentes programas o procesos del sistema.
14. **/usr**: Contiene datos de usuario compartidos, como programas instalados, bibliotecas, y documentación. Se subdivide en directorios como `/usr/bin` (binarios no esenciales), `/usr/lib` (bibliotecas no esenciales), y `/usr/share` (datos compartidos).
15. **/var**: Contiene datos variables, como registros del sistema, archivos temporales, colas de impresión y correos electrónicos.

¿A qué Hace Referencia la Sigla FHS?

La sigla **FHS** significa **Filesystem Hierarchy Standard**. Es un estándar que define la estructura y las convenciones para los directorios y su contenido en sistemas operativos tipo Unix, incluyendo GNU/Linux. Su propósito es asegurar una estructura de archivos consistente y predecible en todas las distribuciones, facilitando la administración del sistema, la escritura de scripts, y el desarrollo de software.

7 Particiones:

▼ (a) Definición. Tipos de particiones. Ventajas y Desventajas.

Definición:

Una **partición** es una división lógica de un disco de almacenamiento físico que permite a los sistemas operativos gestionar el almacenamiento de datos de manera independiente. Las particiones actúan como unidades de

almacenamiento separadas dentro de un mismo disco físico, lo que facilita la organización de los datos y permite la instalación de múltiples sistemas operativos en el mismo disco.

Tipos de Particiones:

1. **Partición Primaria:**
    - **Descripción:** Una partición que se define directamente en la tabla de particiones del disco. Un disco con una tabla de particiones MBR (Master Boot Record) puede tener hasta cuatro particiones primarias.
    - **Ventajas:**
        - Puede ser marcada como "bootable" (de arranque) para instalar y arrancar sistemas operativos.
        - Acceso directo desde el sistema operativo.
    - **Desventajas:**
        - Limitación de hasta cuatro particiones primarias por disco (en esquemas MBR).
2. **Partición Extendida:**
    - **Descripción:** Una partición que puede contener múltiples particiones lógicas. Solo puede haber una partición extendida por disco.
    - **Ventajas:**
        - Permite crear más de cuatro particiones en un disco MBR, ya que dentro de la partición extendida pueden existir varias particiones lógicas.
    - **Desventajas:**
        - No puede ser usada para almacenar datos directamente, solo contiene particiones lógicas.
3. **Partición Lógica:**

- **Descripción:** Particiones creadas dentro de una partición extendida. Las particiones lógicas pueden ser utilizadas para almacenamiento o para instalar sistemas operativos.
- **Ventajas:**
  - Supera la limitación de cuatro particiones en discos MBR permitiendo múltiples particiones adicionales.
- **Desventajas:**
  - Ligera complejidad añadida en la configuración debido a su contención dentro de una partición extendida.

4. **Partición GPT (GUID Partition Table):**
   - **Descripción:** Un esquema de particionamiento más moderno que permite un número mucho mayor de particiones y soporta discos de mayor tamaño.
   - **Ventajas:**
     - No tiene límite en el número de particiones (prácticamente ilimitadas).
     - Soporta discos más grandes (mayores a 2 TB).
     - Incluye redundancia de la tabla de particiones para mayor seguridad.
   - **Desventajas:**
     - No compatible con sistemas más antiguos que solo soportan MBR.

Ventajas y Desventajas de Particionar un Disco:

- **Ventajas:**
  - Mejora la organización de los datos.
  - Facilita la administración de múltiples sistemas operativos en un solo disco.

    - Aumenta la seguridad al aislar sistemas y datos.
    - Puede mejorar el rendimiento al optimizar el sistema de archivos para diferentes tipos de datos.
- **Desventajas:**
    - Si las particiones no están bien dimensionadas, puede haber desperdicio de espacio.
    - La gestión de múltiples particiones puede ser más compleja.
    - Limitaciones del esquema MBR para particionar discos más grandes o con más de cuatro particiones.

▼ (b) ¿Cómo se identifican las particiones en GNU/Linux? (Considere discos IDE, SCSI y
SATA).

En GNU/Linux, las particiones se identifican según el tipo de disco y el sistema de particionamiento.

- **Discos IDE (PATA):** Las unidades IDE se identifican con nombres como `hdX`, donde `X` es una letra que representa el disco (por ejemplo, `hda` para el primer disco IDE, `hdb` para el segundo).
- **Discos SCSI y SATA:** Las unidades SCSI y SATA se identifican con nombres como `sdX`, donde `X` es una letra que representa el disco (por ejemplo, `sda` para el primer disco SCSI/SATA, `sdb` para el segundo).

Identificación de Particiones:

- **Particiones Primarias y Extendidas:** Se numeran del 1 al 4 (por ejemplo, `sda1`, `sda2`, `sda3`, `sda4`).
- **Particiones Lógicas:** Se numeran a partir del 5 (por ejemplo, `sda5`, `sda6`, etc.).

▼ (c) ¿Cuántas particiones son necesarias como mínimo para instalar GNU/Linux? Nómbre
las indicando tipo de partición, identificación, tipo de File System y punto de montaje.

Para instalar GNU/Linux, son necesarias como mínimo dos particiones:

1. **Partición raíz (`/`)**:
    - **Tipo de partición**: Primaria o lógica.
    - **Identificación**: `83` en sistemas con tablas de particiones MBR (Master Boot Record).
    - **Tipo de File System**: Ext4 (u otro como ext3, ext2, XFS, Btrfs).
    - **Punto de montaje**: `/` (raíz del sistema de archivos).
2. **Partición de intercambio (Swap)**:
    - **Tipo de partición**: Primaria o lógica.
    - **Identificación**: `82` en sistemas con tablas de particiones MBR.
    - **Tipo de File System**: No tiene un sistema de archivos tradicional, ya que se utiliza como espacio de intercambio.
    - **Punto de montaje**: No tiene un punto de montaje, ya que se usa directamente por el sistema para intercambio de memoria.

Estas dos particiones son lo mínimo necesario para una instalación funcional de GNU/Linux. La partición raíz contiene todos los archivos del sistema operativo, y la partición de intercambio se usa para gestionar la memoria virtual.

▼

▼ (d) Ejemplifique diversos casos de particionamiento dependiendo del tipo de tarea que se
deba realizar en su sistema operativo.

1. **Servidor web:**
    - `/`**:** (Raíz) – 20 GB.
    - `/var`**:** (Datos de aplicaciones web, logs) – 50 GB.
    - `/home`**:** (Datos de usuarios) – 10 GB.
    - `swap`**:** – Tamaño equivalente a la RAM o 1.5 veces.
2. **Sistema de escritorio general:**

- `/`: (Raíz) – 30 GB.
- `/home`: (Archivos personales) – Resto del espacio disponible.
- `swap`: – Tamaño equivalente a la RAM.

3. **Sistema de pruebas y desarrollo:**
   - `/`: (Raíz) – 50 GB.
   - `/home`: (Datos personales) – 100 GB.
   - `/opt`: (Software adicional y herramientas de desarrollo) – 50 GB.
   - `swap`: – Tamaño equivalente a la RAM o 1.5 veces.

▼ (e) Tipos de Software para Particionar y Comparación

1. **GParted (GNOME Partition Editor):**
   - **Características:** Interfaz gráfica amigable, soporta la mayoría de sistemas de archivos (ext2/3/4, NTFS, FAT32, etc.), permite redimensionar, mover, copiar, y borrar particiones.
   - **Ventajas:** Gratuito, código abierto, muy popular y fácil de usar.
   - **Desventajas:** No soporta particiones encriptadas directamente, y algunas operaciones pueden ser lentas en discos grandes.
2. **fdisk (Fixed Disk):**
   - **Características:** Herramienta de línea de comandos, se usa principalmente para particiones con tabla MBR.
   - **Ventajas:** Muy ligero, disponible en casi todas las distribuciones de Linux.
   - **Desventajas:** Interfaz menos intuitiva, no soporta GPT nativamente (pero sí su contraparte `gdisk`).
3. **parted (GNU Parted):**
   - **Características:** Herramienta de línea de comandos y también disponible con interfaz gráfica (como GParted), soporta MBR y GPT.

- **Ventajas:** Flexible, puede trabajar con particiones grandes y discos de más de 2 TB.
- **Desventajas:** Interfaz menos amigable en la versión de línea de comandos.

4. **KDE Partition Manager:**
   - **Características:** Similar a GParted pero integrado en el entorno de escritorio KDE.
   - **Ventajas:** Buena integración con KDE, fácil de usar.
   - **Desventajas:** Funcionalmente similar a GParted, por lo que no ofrece grandes ventajas si ya se está usando GParted.
5. **cfdisk:**
   - **Características:** Herramienta de línea de comandos con una interfaz semi-gráfica para la gestión de particiones.
   - **Ventajas:** Más fácil de usar que `fdisk`, soporta MBR.
   - **Desventajas:** No tan flexible como `parted` o GParted, no soporta GPT.

Comparación General:

- **Interfaz de Usuario:** GParted y KDE Partition Manager son los más fáciles de usar gracias a sus interfaces gráficas.
- **Soporte de GPT:** `parted` y `gdisk` son preferidos para trabajar con GPT, mientras que `fdisk` y `cfdisk` son limitados a MBR.
- **Versatilidad:** `parted` es muy versátil, ofreciendo soporte para una amplia gama de tareas de particionamiento en discos tanto MBR como GPT.
- **Disponibilidad:** `fdisk` y `parted` suelen estar preinstalados en la mayoría de las distribuciones de Linux, mientras que GParted puede requerir instalación adicional.

8 Arranque (bootstrap) de un Sistema Operativo:

▼ ¿Qué es el BIOS? ¿Qué tarea realiza?

BIOS (Basic Input/Output System) es un firmware que está integrado en la placa base de una computadora y se ejecuta cuando esta se enciende. Sus principales tareas son:

1. **Inicialización del hardware**: Realiza un proceso de autoevaluación conocido como POST (Power-On Self Test) para verificar que todos los componentes básicos de la computadora (como memoria RAM, discos duros, teclado, etc.) funcionan correctamente.
2. **Gestión de la configuración básica del hardware**: Permite a los usuarios configurar opciones básicas de hardware, como el orden de arranque de los dispositivos, el reloj del sistema, y parámetros de funcionamiento de la memoria y procesadores.
3. **Carga del sistema operativo**: Localiza el gestor de arranque en un dispositivo de almacenamiento (como un disco duro o SSD) y lo carga en memoria para iniciar el sistema operativo.

▼ (b) ¿Qué es UEFI? ¿Cuál es su función?

UEFI (Unified Extensible Firmware Interface) es un estándar más moderno que reemplaza al BIOS tradicional. Fue diseñado para superar las limitaciones del BIOS y ofrecer más funcionalidades y seguridad. Sus funciones principales son:

1. **Inicialización del hardware**: Al igual que el BIOS, inicializa el hardware y realiza pruebas POST.
2. **Interfaz gráfica y soporte de ratón**: Proporciona una interfaz más moderna y fácil de usar en comparación con el BIOS.
3. **Compatibilidad con sistemas de archivos**: UEFI puede leer sistemas de archivos directamente, lo que permite cargar archivos desde discos duros y particiones con formatos avanzados como FAT32.
4. **Modo seguro de arranque (Secure Boot)**: Ofrece una capa adicional de seguridad al verificar la firma digital de los sistemas operativos antes de permitir su arranque, protegiendo contra el malware en el proceso de arranque.
5. **Soporte para particiones GPT**: A diferencia del BIOS, UEFI admite el esquema de particiones GPT, que es más moderno y ofrece ventajas

sobre el MBR.

▼ ¿Qué es el MBR? ¿Que es el MBC?

- **MBR** (Master Boot Record): Es el primer sector de un dispositivo de almacenamiento, como un disco duro o una unidad SSD, que contiene la información necesaria para arrancar el sistema operativo. El MBR incluye una tabla de particiones que describe las particiones en el disco y un pequeño fragmento de código ejecutable llamado **MBC** (Master Boot Code).
- **MBC** (Master Boot Code): Es parte del MBR y contiene el código de arranque que el BIOS ejecuta para iniciar el proceso de arranque del sistema operativo. El MBC carga el gestor de arranque o el sistema operativo desde la partición activa.

▼ ¿A qué hacen referencia las siglas GPT? ¿Qué sustituye? Indique cuál es su formato.

GPT (GUID Partition Table) es un estándar para la tabla de particiones en un disco duro. GPT sustituye al **MBR** (Master Boot Record) y ofrece varias ventajas sobre él:

1. **Capacidad**: GPT admite discos de hasta 9.4 ZB (zettabytes), mientras que MBR se limita a 2 TB.
2. **Número de particiones**: GPT permite un número prácticamente ilimitado de particiones (aunque las especificaciones de UEFI generalmente limitan a 128), mientras que MBR sólo permite 4 particiones primarias.
3. **Redundancia y recuperación**: GPT almacena múltiples copias de los datos de la tabla de particiones para mejorar la integridad de los datos y ofrece una CRC32 para verificar la integridad de estos datos.
4. **Formato**: GPT utiliza identificadores únicos globales (GUID) para las particiones, asegurando que cada partición tiene un identificador único.

▼ ¿Cuál es la funcionalidad de un “Gestor de Arranque”? ¿Qué tipos existen? ¿Dónde se
instalan? Cite gestores de arranque conocidos.

Un **gestor de arranque** es un software que se ejecuta durante el proceso de arranque del sistema para cargar y transferir el control al sistema operativo. Su funcionalidad principal es permitir que el usuario seleccione entre diferentes sistemas operativos instalados o configuraciones de arranque.

- **Tipos**:
    1. **Gestores de arranque de sistema único**: Arrancan directamente un solo sistema operativo.
    2. **Gestores de arranque múltiples**: Permiten seleccionar entre múltiples sistemas operativos instalados en la computadora.
- **Instalación**: Los gestores de arranque suelen instalarse en el MBR del disco o en la partición EFI si se utiliza UEFI.
- **Gestores de arranque conocidos**:
    - **GRUB** (GRand Unified Bootloader): Utilizado comúnmente en sistemas GNU/Linux.
    - **LILO** (Linux Loader): Otro cargador de arranque para sistemas Linux, aunque menos común actualmente.
    - **Windows Boot Manager**: Utilizado en sistemas Windows.
    - **rEFInd**: Un gestor de arranque gráfico para sistemas UEFI, soportando múltiples sistemas operativos.

▼ ¿Cuáles son los pasos que se suceden desde que se prende una computadora hasta que
el Sistema Operativo es cargado (proceso de bootstrap)?

- **1 Encendido del hardware**: El usuario enciende la computadora y la fuente de alimentación proporciona energía a la placa base.
- **2 POST (Power-On Self Test)**: El BIOS/UEFI realiza una serie de pruebas para asegurarse de que los componentes básicos del hardware funcionan correctamente.
- **3 Carga del gestor de arranque**: El BIOS/UEFI busca el gestor de arranque en el primer dispositivo de almacenamiento disponible (según

la configuración de arranque).

- **4 Ejecución del gestor de arranque**: El gestor de arranque se carga en memoria y se ejecuta, permitiendo al usuario seleccionar el sistema operativo que desea arrancar.
- **5 Carga del kernel del sistema operativo**: El gestor de arranque carga el núcleo del sistema operativo en memoria.
- **6 Inicio del sistema operativo**: El sistema operativo toma el control del hardware y comienza el proceso de inicialización del sistema, como cargar controladores y servicios, hasta que se presenta la interfaz de usuario.

▼ Analice el proceso de arranque en GNU/Linux y describa sus principales pasos.

- **1 POST**: Igual que en cualquier otro sistema, el BIOS/UEFI realiza el POST.
- **2 Gestor de arranque**: Se carga el gestor de arranque (por ejemplo, GRUB) que presenta un menú para seleccionar el sistema operativo o kernel a arrancar.
- **3 Cargar kernel y initramfs**: El gestor de arranque carga en memoria el kernel de Linux y el archivo initramfs (initial RAM filesystem), que contiene los controladores necesarios para montar el sistema de archivos raíz.
- **4 Inicio del kernel**: El kernel de Linux se inicializa y configura el hardware básico.
- **5 Ejecutar init**: El kernel busca y ejecuta el proceso `init`, que es el primer proceso en ejecutarse en un sistema GNU/Linux. Dependiendo de la distribución y configuración, `init` puede ser el clásico System V, Upstart, o systemd.
- **6 Inicialización del sistema**: `init` o `systemd` ejecuta scripts y servicios para configurar el entorno de usuario y preparar el sistema para su uso, montando sistemas de archivos, configurando red, etc.

- **7 Inicio de sesión**: Finalmente, se presenta la pantalla de inicio de sesión (login) o el entorno de escritorio si se configura un inicio de sesión automático.

▼ ¿Cuáles son los pasos que se suceden en el proceso de parada (shutdown) de GNU/Li
nux?

- **1 Solicitud de apagado**: El usuario solicita apagar el sistema mediante un comando o interfaz gráfica (`shutdown`, `halt`, `poweroff` o `reboot`).
- **2 Envío de señales**: El proceso de apagado envía señales a todos los procesos en ejecución para que se terminen de forma ordenada.
- **3 Finalización de procesos**: Los procesos terminan sus tareas y se cierran.
- **4 Desmontaje de sistemas de archivos**: El sistema desmonta todos los sistemas de archivos montados para evitar corrupción de datos.
- **5 Sincronización de discos**: Se sincronizan los discos para asegurarse de que todos los datos en caché se escriban en disco.
- **6 Apagado del hardware**: Finalmente, el sistema envía señales para apagar la energía del hardware.

▼ ¿Es posible tener en una PC GNU/Linux y otro Sistema Operativo instalado? Justifi
que.

Sí, es posible tener instalados GNU/Linux y otro sistema operativo (como Windows) en la misma PC. Esto se llama "arranque dual" o "dual-boot".

Justificación:

1. **Soporte de arranque múltiple**: Los gestores de arranque como GRUB pueden gestionar varios sistemas operativos, permitiendo al usuario elegir cuál arrancar durante el proceso de inicio.
2. **Particiones separadas**: Los discos duros pueden dividirse en múltiples particiones, permitiendo que cada sistema operativo tenga su propio espacio de almacenamiento aislado del otro.

3. **Compatibilidad de hardware**: La mayoría de las PCs actuales son compatibles con GNU/Linux y otros sistemas operativos, facilitando la instalación y funcionamiento de ambos en la misma máquina.

9. Archivos:

▼ ¿Cómo se identifican los archivos en GNU/Linux?

En GNU/Linux, los archivos se identifican principalmente por su nombre y su ubicación dentro del sistema de archivos. Sin embargo, más allá del nombre, GNU/Linux utiliza una serie de atributos para identificar y manejar los archivos:

1. **Nombre del archivo**: Cada archivo tiene un nombre que lo identifica dentro de un directorio. Los nombres de archivo pueden contener casi cualquier carácter, excepto `/` y el carácter nulo (`\0`).
2. **Inodo**: Cada archivo en un sistema de archivos ext2/ext3/ext4 (y otros) está asociado a un inodo (inode). Un inodo es una estructura de datos que almacena información sobre un archivo, como permisos, propietario, tamaño, tipo, y ubicaciones del disco donde se encuentran sus datos. El inodo no contiene el nombre del archivo, sino que este es almacenado en la entrada del directorio que contiene el archivo.
3. **Permisos y propietario**: Cada archivo tiene permisos asociados que determinan quién puede leer, escribir o ejecutar el archivo. También tienen un propietario y un grupo asociado.
4. **Enlaces duros y simbólicos**: Los archivos pueden tener múltiples enlaces duros (hard links), que son diferentes nombres en el sistema de archivos que apuntan al mismo inodo. Los enlaces simbólicos (symlinks) son referencias que apuntan a otro archivo por su nombre.
5. **Metadatos**: Atributos adicionales como la fecha de creación, modificación y acceso, así como otros atributos extendidos.

▼ Investigue el funcionamiento de los editores vi y mcedit, y los comandos cat y more.

Editores:

1. **vi**:

- **Descripción**: `vi` (Visual Editor) es un editor de texto modal muy potente y ampliamente utilizado en sistemas UNIX y GNU/Linux.
- **Funcionamiento**:
  - **Modo de comandos**: Se utiliza para navegar por el texto, eliminar, copiar y pegar texto, y ejecutar comandos.
  - **Modo de inserción**: Se utiliza para insertar o modificar texto.
- **Comandos básicos**:
  - `i`: Entra en modo de inserción antes del cursor.
  - `Esc`: Vuelve al modo de comandos.
  - `:w`: Guarda el archivo.
  - `:q`: Sale del editor.
  - `:wq`: Guarda y sale.
  - `:q!`: Sale sin guardar.

2. **mcedit**:
   - **Descripción**: `mcedit` es el editor de texto integrado en `Midnight Commander` (MC), un administrador de archivos de interfaz gráfica para la terminal.
   - **Funcionamiento**:
     - Es similar a otros editores de texto en terminal, con un diseño de interfaz fácil de usar que muestra un menú con las opciones más comunes en la parte inferior de la pantalla.
     - Los comandos se pueden ejecutar utilizando combinaciones de teclas como `F2` para guardar, `F10` para salir, etc.
   - **Comandos básicos**:
     - `F2`: Guardar.
     - `F10`: Salir.
     - `F5`: Buscar texto.
     - `F6`: Buscar y reemplazar.

Comandos:

1. **cat**:
    - **Descripción**: `cat` (concatenate) es un comando utilizado para concatenar y mostrar el contenido de archivos en la terminal.
    - **Uso básico**:
        - `cat archivo.txt`: Muestra el contenido de `archivo.txt`.
        - `cat archivo1.txt archivo2.txt > combinado.txt`: Combina los contenidos de `archivo1.txt` y `archivo2.txt` y los guarda en `combinado.txt`.
    - **Funcionalidad**: Es útil para visualizar archivos cortos, combinar archivos y redirigir el contenido de archivos.
2. **more**:
    - **Descripción**: `more` es un comando utilizado para visualizar el contenido de un archivo página por página en la terminal.
    - **Uso básico**:
        - `more archivo.txt`: Muestra el contenido de `archivo.txt` una página a la vez.
    - **Funcionalidad**: Ideal para leer archivos largos en la terminal. Permite avanzar una pantalla (presionando la barra espaciadora) o una línea (presionando Enter).

▼ Cree un archivo llamado “prueba.exe” en su directorio personal usando el vi. El mismo
debe contener su número de alumno y su nombre.

Para crear un archivo llamado `prueba.exe` en tu directorio personal usando `vi`, puedes seguir estos pasos en la terminal:

1. Abre el editor `vi` con el nombre del archivo:

```
bashCopiar código
vi ~/prueba.exe
```

```

```

2. Entra en el modo de inserción presionando `i`.
3. Escribe tu número de alumno y tu nombre, por ejemplo:

```
yamlCopiar código
Número de alumno: 12345678
Nombre: Juan Pérez
```

4. Presiona `Esc` para salir del modo de inserción.
5. Guarda el archivo y sal de `vi` escribiendo `:wq` y presionando Enter.

▼ Investigue el funcionamiento del comando file. Pruébelo con diferentes archivos. ¿Qué
diferencia nota?

El comando `file` se utiliza en GNU/Linux para determinar el tipo de un archivo. A diferencia de los sistemas operativos que dependen de las extensiones de archivo para identificar el tipo de archivo, `file` analiza el contenido del archivo para identificar su tipo, basándose en una base de datos de patrones.

Funcionamiento del comando `file`:

- **Uso básico**:
  - `file nombre_de_archivo`: Muestra el tipo de archivo de `nombre_de_archivo`.
- **Ejemplos**:
  - `file prueba.exe`: Podría mostrar `ASCII text` o similar, dependiendo del contenido del archivo y no del nombre o extensión.
  - `file imagen.jpg`: Muestra algo como `JPEG image data, JFIF standard`.
  - `file script.sh`: Podría mostrar `Bourne-Again shell script, ASCII text executable`.

- `file binario`: Muestra algo como `ELF 64-bit LSB executable, x86-64`.

Diferencia que se nota al usar `file` con diferentes archivos:

- **Basado en contenido, no en extensión**: A diferencia de otros sistemas que dependen de la extensión del archivo (como .txt o .jpg), `file` no se basa en la extensión del archivo para identificarlo. Por ejemplo, un archivo llamado `prueba.exe` podría ser identificado como `ASCII text` si contiene solo texto, sin importar la extensión `.exe`.
- **Detección de formatos**: `file` es capaz de detectar una amplia variedad de formatos de archivo, incluso si el archivo no tiene ninguna extensión o tiene una extensión incorrecta. Esto es útil para verificar archivos recibidos o descargados para asegurarse de que son lo que parecen ser.
- **Detalles adicionales**: `file` también proporciona información detallada sobre algunos tipos de archivos, como scripts (mostrando el tipo de script y el intérprete asociado) o ejecutables (mostrando si son de 32 o 64 bits, el tipo de arquitectura, etc.).

10 Indique qué comando es necesario utilizar para realizar cada una de las siguientes acciones.
Investigue su funcionamiento y parámetros más importantes:

▼ data

(a) Cree la carpeta ISO2017

Comando: `mkdir`

- **Uso básico**:

```
bashCopiar código
mkdir ISO2017
```

- **Descripción**: `mkdir` (make directory) se utiliza para crear un nuevo directorio.
- **Parámetros importantes**:
  - `p`: Crea directorios de manera recursiva. Por ejemplo, `mkdir -p dir1/dir2` crea `dir1` y luego `dir2` dentro de `dir1`.
  - `v`: Muestra un mensaje por cada directorio creado.

(b) Acceda a la carpeta (cd)

Comando: `cd`

- **Uso básico**:

```
bashCopiar código
cd ISO2017
```

- **Descripción**: `cd` (change directory) se utiliza para cambiar el directorio actual de trabajo a otro directorio especificado.
- **Parámetros importantes**:
  - `..`: Cambia al directorio padre del actual. Por ejemplo, `cd ..`.
  - `-`: Cambia al directorio anterior.

(c) Cree dos archivos con los nombres iso2017-1 e iso2017-2 (touch)

Comando: `touch`

- **Uso básico**:

```
bashCopiar código
touch iso2017-1 iso2017-2
```

- **Descripción**: `touch` se utiliza para crear archivos vacíos o actualizar la marca de tiempo de acceso y modificación de un archivo existente.

- **Parámetros importantes**:
  - `a`: Cambia solo la hora de acceso.
  - `m`: Cambia solo la hora de modificación.
  - `t`: Establece una fecha/hora específica en formato [[CC]YY]MMDDhhmm[.ss].

(d) Liste el contenido del directorio actual (ls)

Comando: `ls`

- **Uso básico**:

```
bashCopiar código
ls
```

- **Descripción**: `ls` lista los archivos y directorios en el directorio actual o especificado.
- **Parámetros importantes**:
  - `l`: Muestra el listado en formato largo (con detalles como permisos, número de enlaces, propietario, tamaño y fecha de modificación).
  - `a`: Muestra todos los archivos, incluidos los ocultos (que empiezan con `.`).
  - `h`: (con `l`) Muestra tamaños de archivos en formato legible para humanos (KB, MB, GB, etc.).

(e) Visualizar la ruta donde estoy situado (pwd)

Comando: `pwd`

- **Uso básico**:

```
bashCopiar código
pwd
```

- **Descripción**: `pwd` (print working directory) muestra la ruta completa del directorio actual de trabajo.
- **Parámetros importantes**:
  - `P`: Muestra la ruta física, sin enlaces simbólicos.
  - `L`: Muestra la ruta lógica (incluye enlaces simbólicos, es el comportamiento predeterminado).

(f) Busque todos los archivos en los que su nombre contiene la cadena “iso*” (find)

Comando: `find`

- **Uso básico**:

```
bashCopiar código
find . -name "iso*"
```

- **Descripción**: `find` busca archivos y directorios en una jerarquía de directorios según criterios específicos.
- **Parámetros importantes**:
  - `name pattern`: Busca archivos que coincidan con el patrón especificado (sensibles a mayúsculas y minúsculas).
  - `iname pattern`: Busca archivos que coincidan con el patrón especificado (insensibles a mayúsculas y minúsculas).
  - `type type`: Especifica el tipo de archivo a buscar (`f` para archivos regulares, `d` para directorios).
  - `size n`: Busca archivos de un tamaño específico.
  - `exec command {} \;`: Ejecuta un comando en cada archivo encontrado.

(g) Informar la cantidad de espacio libre en disco (df)

Comando: `df`

- **Uso básico**:

```
bashCopiar código
df
```

- **Descripción**: `df` (disk free) muestra información sobre el uso del espacio en disco de todos los sistemas de archivos montados.
- **Parámetros importantes**:
  - `h`: Muestra tamaños en formato legible para humanos (KB, MB, GB).
  - `T`: Muestra el tipo de sistema de archivos.
  - `i`: Muestra el uso de inodos en lugar del espacio en disco.

(h) Verifique los usuarios conectados al sistema (who)

Comando: `who`

- **Uso básico**:

```
bashCopiar código
who
```

- **Descripción**: `who` muestra una lista de usuarios que actualmente tienen sesiones iniciadas en el sistema.
- **Parámetros importantes**:
  - `H`: Muestra encabezados para cada columna de la salida.
  - `u`: Muestra información sobre la hora de inicio de sesión y el tiempo de inactividad.
  - `q`: Muestra una lista de usuarios y el número de usuarios conectados.

(i) Acceder al archivo iso2017-1 e ingresar Nombre y Apellido

Comando: `vi`

- **Uso básico**:

```
bashCopiar código
vi iso2017-1
```

- **Descripción**: `vi` es un editor de texto en modo terminal. Puedes usarlo para editar archivos como `iso2017-1`.
- **Pasos para ingresar Nombre y Apellido**:
  1. Abre el archivo con `vi iso2017-1`.
  2. Presiona `i` para entrar en el modo de inserción.
  3. Escribe tu nombre y apellido.
  4. Presiona `Esc` para salir del modo de inserción.
  5. Guarda y cierra el archivo escribiendo `:wq` y presionando Enter.

(j) Mostrar en pantalla las últimas líneas de un archivo (tail)

Comando: `tail`

- **Uso básico**:

```
bashCopiar código
tail archivo.txt
```

- **Descripción**: `tail` muestra las últimas líneas de un archivo.
- **Parámetros importantes**:
  - `n N`: Muestra las últimas `N` líneas del archivo. Por defecto, muestra las últimas 10 líneas.
  - `f`: Muestra en tiempo real las líneas que se agregan al final de un archivo (útil para monitorear logs).

11 Investigue su funcionamiento y parámetros más importantes

▼ data

(a) `shutdown`

Función: `shutdown` se utiliza para apagar, reiniciar o detener el sistema de manera controlada.

- **Uso básico**:

```
bashCopiar código
shutdown [opciones] [tiempo] [mensaje]
```

- **Parámetros importantes**:
  - `h`: Apaga el sistema.
  - `r`: Reinicia el sistema.
  - `c`: Cancela un apagado o reinicio programado.
  - `P`: Apaga el sistema y apaga la energía (en la mayoría de las distribuciones, `h` hace esto por defecto).
  - `H`: Apaga el sistema sin apagar la energía (pone el sistema en modo de espera).
  - `time`: Especifica cuándo debe ocurrir el apagado. Puede ser `now` para inmediato o un tiempo en minutos (`+5` para 5 minutos más tarde).
  - `mensaje`: Muestra un mensaje de aviso a los usuarios conectados.

(b) `reboot`

Función: `reboot` se utiliza para reiniciar el sistema.

- **Uso básico**:

```
bashCopiar código
reboot
```

```

- **Parámetros importantes**:
  - `f`: Fuerza el reinicio sin llamar a `shutdown`.
  - `p`: Apaga el sistema en lugar de reiniciarlo.
  - `w`: Simula el reinicio sin realmente llevarlo a cabo (solo para pruebas).

## (c) `halt`

**Función**: `halt` se utiliza para detener el sistema de manera inmediata.

- **Uso básico**:

```
bashCopiar código
halt
```

- **Parámetros importantes**:
  - `f`: Forza la parada sin llamar a `shutdown`.
  - `-poweroff`: Detiene el sistema y apaga la energía.
  - `-reboot`: Reinicia el sistema en lugar de detenerlo.

## (d) `locate`

**Función**: `locate` se utiliza para encontrar archivos por nombre en el sistema de archivos.

- **Uso básico**:

```
bashCopiar código
locate [nombre_del_archivo]
```

- **Parámetros importantes**:
  - `i`: Realiza una búsqueda insensible a mayúsculas.
```

- `r`: Permite usar expresiones regulares.
- `c`: Muestra el número de resultados en lugar de los resultados en sí.
- `e`: Solo muestra archivos que existen en el momento de la búsqueda.

(e) `uname`

Función: `uname` muestra información sobre el sistema.

- **Uso básico**:

```
bashCopiar código
uname [opciones]
```

- **Parámetros importantes**:
  - `a`: Muestra toda la información disponible.
  - `s`: Muestra el nombre del sistema operativo.
  - `n`: Muestra el nombre de red del host.
  - `r`: Muestra la versión del kernel.
  - `v`: Muestra la versión del sistema operativo.
  - `m`: Muestra la arquitectura del hardware de la máquina.
  - `p`: Muestra el tipo de procesador (a veces no disponible).
  - `i`: Muestra la plataforma de hardware (a veces no disponible).
  - `o`: Muestra el sistema operativo.

(f) `dmesg`

Función: `dmesg` muestra el buffer de mensajes del kernel.

- **Uso básico**:

```
bashCopiar código
dmesg
```

```

- **Parámetros importantes**:
  - `C`: Limpia el buffer del kernel.
  - `c`: Muestra el buffer y luego lo limpia.
  - `H`: Muestra la salida en un formato legible para humanos (tiempos relativos).
  - `T`: Muestra las marcas de tiempo legibles para humanos.
  - `l`: Filtra los mensajes por nivel de prioridad.

## (g) `lspci`

**Función**: `lspci` muestra información sobre todos los dispositivos PCI (Peripheral Component Interconnect) en el sistema.

- **Uso básico**:

```
bashCopiar código
lspci
```

- **Parámetros importantes**:
  - `v`: Muestra información detallada.
  - `vv` o `vvv`: Muestra aún más detalles.
  - `k`: Muestra los controladores de kernel y módulos utilizados por cada dispositivo.
  - `n`: Muestra los números de ID de dispositivo y vendedor en lugar de sus nombres.
  - `s [dominio]:[bus]:[ranura].[función]`: Muestra información solo para un dispositivo específico.

## (h) `at`
```

Función: `at` programa comandos para que se ejecuten en un momento específico en el futuro.

- **Uso básico**:

```
bashCopiar código
at [hora]
```

- **Parámetros importantes**:
  - `l` o `atq`: Lista los trabajos programados para `at`.
  - `d` o `atrm`: Elimina un trabajo programado.
  - `m`: Envía un correo al usuario cuando el trabajo se complete.
  - `f [archivo]`: Lee los comandos a ejecutar desde un archivo en lugar de la entrada estándar.

(i) `netstat`

Función: `netstat` muestra varias estadísticas de red.

- **Uso básico**:

```
bashCopiar código
netstat [opciones]
```

- **Parámetros importantes**:
  - `a`: Muestra todas las conexiones y puertos de escucha.
  - `t`: Muestra solo las conexiones TCP.
  - `u`: Muestra solo las conexiones UDP.
  - `n`: Muestra las direcciones IP y números de puertos en formato numérico.
  - `p`: Muestra el ID del proceso y el nombre del programa al que pertenece cada conexión.

- `r`: Muestra la tabla de enrutamiento.

(j) `mount`

Función: `mount` monta un sistema de archivos en un directorio.

- **Uso básico**:

```
bashCopiar código
mount [opciones] [dispositivo] [punto_de_montaje]
```

- **Parámetros importantes**:
  - `a`: Monta todos los sistemas de archivos definidos en `/etc/fstab`.
  - `t [tipo]`: Especifica el tipo de sistema de archivos (por ejemplo, `ext4`, `ntfs`).
  - `o [opciones]`: Especifica opciones de montaje adicionales (por ejemplo, `rw` para lectura/escritura, `ro` para solo lectura, `noexec` para no ejecutar archivos).

(k) `umount`

Función: `umount` desmonta un sistema de archivos previamente montado.

- **Uso básico**:

```
bashCopiar código
umount [opciones] [punto_de_montaje|dispositivo]
```

- **Parámetros importantes**:
  - `a`: Desmonta todos los sistemas de archivos que se pueden desmontar.
  - `f`: Fuerza el desmontaje (útil para sistemas de archivos de red).
  - `l`: Desmonta de forma perezosa (lazy), lo que desconecta el sistema de archivos ahora pero espera hasta que deje de estar en uso para

completar el desmontaje.

(l) `head`

Función: `head` muestra las primeras líneas de un archivo.

- **Uso básico**:

```
bashCopiar código
head [opciones] [archivo]
```

- **Parámetros importantes**:
  - `n [N]`: Muestra las primeras `N` líneas del archivo (por defecto muestra 10).
  - `c [N]`: Muestra los primeros `N` bytes del archivo.

(m) `losetup`

Función: `losetup` configura o visualiza dispositivos de bucle (loop devices).

- **Uso básico**:

```
bashCopiar código
losetup [opciones] [dispositivo] [archivo]
```

- **Parámetros importantes**:
  - `a`: Muestra todos los dispositivos de bucle configurados.
  - `f`: Encuentra el primer dispositivo de bucle no utilizado y lo asigna.
  - `d [dispositivo]`: Desasocia un dispositivo de bucle específico.
  - `o [offset]`: Especifica un desplazamiento en bytes dentro del archivo para el dispositivo de bucle.
  - `P`: Analiza automáticamente las particiones de un dispositivo de bucle.

(n) `write`

Función: `write` envía un mensaje a otro usuario que está conectado al mismo sistema.

- **Uso básico**:

```
bashCopiar código
write [usuario] [tty]
```

- **Parámetros importantes**:
  - `usuario`: Especifica el nombre del usuario al que enviar el mensaje.
  - `tty`: Especifica el terminal del usuario si está conectado a más de uno.

(ñ) `mkfs`

Función: `mkfs` (make filesystem) se utiliza para crear un sistema de archivos en un dispositivo.

- **Uso básico**

(o) `fdisk`

Función: `fdisk` es una utilidad de manipulación de particiones para discos duros. Se utiliza principalmente para crear, eliminar, redimensionar, modificar y gestionar las particiones del disco.

Uso básico:

```
bashCopiar código
fdisk [opciones] [dispositivo]
```

- **Advertencia**: El uso de `fdisk` puede afectar directamente al sistema de archivos y los datos almacenados en el disco. Cualquier modificación realizada con este comando puede resultar en la pérdida de datos. Se recomienda hacer copias de seguridad antes de manipular particiones con `fdisk`.

Parámetros importantes:

- **Sin opciones**: Si ejecutas `fdisk` sin opciones, se abrirá una interfaz interactiva para trabajar con el dispositivo especificado.
- **Opciones**:
  - `l`: Lista las tablas de particiones de todos los dispositivos. Por ejemplo, `fdisk -l`.
  - `b [número]`: Define el tamaño del sector en bytes. Por ejemplo, `fdisk -b 4096`.
  - `c [modo]`: Define el modo de compatibilidad de cilindros. Usar `c` con `dos` o `c` con `cylinders`.
  - `u [unidad]`: Define las unidades de tamaño, ya sea `cylinders` o `sectors`. Por ejemplo, `fdisk -u sectors`.

Operaciones interactivas: Una vez dentro del modo interactivo de `fdisk`, puedes utilizar diferentes comandos:

- **m**: Muestra la lista de comandos disponibles.
- **p**: Muestra la tabla de particiones actual del disco.
- **n**: Crea una nueva partición.
- **d**: Elimina una partición existente.
- **q**: Sale sin guardar los cambios.
- **w**: Guarda los cambios y sale.

Ejemplo de uso:

```
bashCopiar código
fdisk /dev/sda
```

- Este comando abrirá `fdisk` en modo interactivo para el disco `/dev/sda`. Desde allí, puedes listar particiones, crear nuevas, eliminar existentes y más.

Recomendaciones:

- Siempre verifica en qué disco estás trabajando para evitar alterar el disco equivocado.
- Realiza operaciones en discos solo si estás seguro de lo que estás haciendo.
- Considera utilizar herramientas más amigables para la manipulación de particiones, como `parted` o `gparted`, si no estás familiarizado con `fdisk`.

12 Investigue su funcionamiento y parámetros más importantes:

▼ Indique en qué directorios se almacenan los comandos mencionados en el ejercicio
anterior.

En sistemas GNU/Linux, los comandos que se mencionan en el ejercicio anterior suelen ser ejecutables que se almacenan en directorios específicos dentro del sistema de archivos. Estos directorios son parte de la variable de entorno `PATH`, que el sistema utiliza para buscar los comandos cuando son ejecutados.

Los directorios más comunes donde se almacenan los comandos en GNU/Linux son:

1. `/bin`: Contiene comandos esenciales para todos los usuarios. Aquí se encuentran utilidades básicas necesarias para la operación del sistema, como `ls`, `cp`, `mv`, `rm`, `mkdir`, entre otros.
2. `/sbin`: Contiene comandos esenciales de administración del sistema. Solo los usuarios con privilegios administrativos (como el superusuario o root) necesitan acceder a estos comandos. Aquí se encuentran comandos como `shutdown`, `reboot`, `fdisk`, `mkfs`, etc.
3. `/usr/bin`: Contiene la mayoría de los comandos de usuario no esenciales. Este directorio suele contener programas y comandos de aplicación que no son necesarios para el arranque básico del sistema, pero que son útiles para el usuario. Ejemplos incluyen `vi`, `locate`, `write`, `tail`, `head`, `at`, `netstat`, etc.

4. `/usr/sbin`: Similar a `/usr/bin`, pero para binarios de administración de sistema que no son esenciales para el arranque. Aquí se encuentran comandos adicionales que pueden ser usados por el administrador del sistema, como `lspci`, `losetup`, entre otros.
5. `/usr/local/bin` **y** `/usr/local/sbin`: Estos directorios son utilizados para almacenar programas y comandos que son instalados localmente por el administrador del sistema, y que no forman parte de la instalación estándar del sistema.

Para averiguar exactamente dónde se encuentra un comando específico, puedes usar el comando `which` seguido del nombre del comando, como se muestra a continuación:

```
bashCopiar código
which [nombre_del_comando]
```

Por ejemplo:

- Para encontrar dónde se encuentra `ls`, puedes ejecutar `which ls` y obtendrás `/bin/ls`.
- Para `fdisk`, puedes ejecutar `which fdisk` y obtendrás `/sbin/fdisk`.

Este comando busca el ejecutable en los directorios listados en la variable de entorno `PATH` y te proporciona la ruta completa del comando.