* 1)
  + **GNU/Linux**:
    - Es multiusuario: permite que múltiples usuarios accedan y utilicen el sistema al mismo tiempo sin interferir unos con otros. Cada usuario tiene su propio entorno, configuración y permisos de acceso.
    - Es multitarea y multiprocesador: puede ejecutar múltiples tareas o procesos al mismo tiempo y está diseñado para aprovechar al máximo las capacidades de sistemas con múltiples procesadores (o núcleos de procesador), distribuyendo las tareas entre ellos para mejorar el rendimiento y la eficiencia.
    - Es altamente portable: es un sistema operativo que puede ejecutarse en una amplia variedad de hardware, desde supercomputadoras hasta teléfonos móviles, en diferentes arquitecturas de CPU (como x86, ARM, etc.).
    - Posee diversos intérpretes de comandos, de los cuales algunos son programables: ofrece una variedad de intérpretes de comandos, también conocidos como shells (como Bash, Zsh, Fish, etc.), que permiten a los usuarios interactuar con el sistema mediante líneas de comando. Algunos de estos shells son programables, lo que significa que los usuarios pueden escribir scripts (programas) para automatizar tareas y personalizar su entorno de trabajo.
    - Permite el manejo de usuarios y permisos: tiene un sistema robusto para manejar usuarios y permisos. Cada archivo y directorio en el sistema tiene permisos que determinan quién puede leer, escribir o ejecutar el archivo.
    - Todo es un archivo (hasta los dispositivos y directorios): todo se representa como un archivo en el sistema, desde archivos de texto hasta dispositivos de hardware (como discos duros o impresoras). Por ejemplo, los discos duros se representan como archivos en el directorio /dev.
    - Cada directorio puede estar en una partición diferente (/temp, /home, etc.): permite que diferentes directorios del sistema de archivos estén montados en diferentes particiones o incluso en diferentes discos físicos. Esto proporciona flexibilidad en la gestión de datos y almacenamiento.
    - Es case sensitive: distingue entre mayúsculas y minúsculas en los nombres de archivos y comandos.
    - Es código abierto: es un sistema operativo de código abierto, lo que significa que su código fuente está disponible para que cualquier persona lo vea, modifique y distribuya.
  + **Windows**:
    - Multiusuario: Aunque Windows permite múltiples cuentas de usuario, no soporta múltiples usuarios simultáneamente en la mayoría de las versiones de escritorio.
    - Multitarea y Multiprocesador: Windows también soporta multitarea y es compatible con sistemas multiprocesador, distribuyendo las tareas entre los núcleos disponibles para maximizar el rendimiento.
    - Altamente Portable: Windows es menos portable que GNU/Linux, ya que está diseñado principalmente para arquitecturas x86 y ARM, y está estrechamente vinculado al hardware específico.
    - Intérpretes de Comandos: Windows incluye el símbolo del sistema (CMD) y PowerShell, que es programable y más avanzado que CMD, pero ambos son menos poderosos y flexibles que los shells de GNU/Linux.
    - Manejo de Usuarios y Permisos: Windows tiene un sistema de permisos de archivos, pero es menos granular y flexible que el de GNU/Linux. Los permisos se gestionan a través del sistema de archivos NTFS.
    - Todo es un Archivo: Windows no sigue estrictamente la filosofía de "todo es un archivo". Los dispositivos y otros recursos del sistema se manejan a menudo a través de interfaces específicas en lugar de archivos del sistema.
    - Directorios en Diferentes Particiones: Windows permite montar diferentes unidades en directorios específicos, pero no es tan común como en GNU/Linux. Las particiones suelen tener sus propias letras de unidad (C:, D:, etc.).
    - Case Sensitive: Windows no distingue entre mayúsculas y minúsculas en nombres de archivos de manera predeterminada.
    - Código Abierto: Windows es un sistema operativo propietario, lo que significa que su código fuente no está disponible para el público y no puede ser modificado ni redistribuido libremente.
  + **GNU**: es un proyecto de software libre iniciado por Richard Stallman en 1983 con el objetivo de crear un sistema operativo completamente libre, similar a Unix, pero compuesto enteramente de software que los usuarios pudieran usar, modificar y distribuir libremente. El nombre "GNU" es un acrónimo recursivo que significa "GNU's Not Unix" (GNU No es Unix).
  + En 1983, Richard Stallman, un programador y activista del software libre, inició el proyecto GNU con el objetivo de crear un sistema operativo completo y libre, similar a Unix. Unix era un sistema operativo muy popular en ese momento, pero era propietario. Stallman quería desarrollar una alternativa que fuera completamente libre, permitiendo a cualquier persona usar, modificar y distribuir el software sin restricciones.

Para garantizar que todo el software desarrollado bajo el proyecto GNU permaneciera libre, Stallman creó la Licencia Pública General de GNU (GNU GPL). Esta licencia es un "marco regulatorio" que asegura que cualquier software cubierto por ella puede ser utilizado, modificado y redistribuido libremente, pero también obliga a que cualquier derivado de ese software se distribuya bajo la misma licencia. Esto garantiza que el software y sus modificaciones sigan siendo libres para todos.

En 1985, Richard Stallman fundó la Free Software Foundation (FSF), una organización dedicada a apoyar y promover el software libre, incluida la financiación y el desarrollo del proyecto GNU. Para 1990, el proyecto GNU había desarrollado varias herramientas esenciales para un sistema operativo. Esto incluía GNU Emacs, un poderoso editor de textos; el compilador GNU (GCC), que permite traducir código fuente en ejecutables; y muchas bibliotecas de software necesarias para construir y ejecutar programas, características típicas de un sistema Unix.

Aunque GNU tenía muchas herramientas y programas básicos, aún carecía de un componente crucial: el núcleo (o kernel). El equipo de GNU había estado trabajando en un kernel llamado TRIX, pero decidieron abandonar este proyecto en 1988 porque era demasiado complejo y solo funcionaba en hardware muy caro, lo que no era viable para un sistema operativo que pretendía ser libre y accesible. Tras abandonar TRIX, el proyecto GNU optó por utilizar el núcleo MACH como base para desarrollar su propio kernel, llamado GNU Hurd. Sin embargo, el desarrollo de Hurd fue muy complicado y lento, y nunca alcanzó un estado completamente funcional. En 1991, un estudiante finlandés llamado Linus Torvalds comenzó a desarrollar su propio kernel, llamado Linux. A diferencia de GNU Hurd, Linux fue rápidamente adoptado por la comunidad de desarrolladores y distribuido bajo la GPL, lo que lo hizo compatible con los principios del software libre. En 1992, se decidió combinar el kernel Linux de Linus Torvalds con las herramientas y software del proyecto GNU, creando un sistema operativo completo y funcional. Esta combinación es lo que hoy conocemos como GNU/Linux.

* + **Multitarea** es la capacidad de un sistema operativo para ejecutar múltiples procesos o tareas de manera simultánea o al mismo tiempo. En un entorno multitarea, un sistema operativo puede gestionar y coordinar varias aplicaciones o procesos de usuario y del sistema para que parezca que todos se están ejecutando al mismo tiempo, incluso en un sistema con un solo procesador. **GNU/Linux** es un sistema operativo que soporta multitarea de manera eficiente. Desde sus primeras versiones, Linux ha sido diseñado para ser un sistema operativo multitarea, capaz de manejar múltiples procesos y usuarios simultáneamente.
  + **POSIX** (Portable Operating System Interface) es un conjunto de estándares que definen las interfaces de programación (APIs), comandos y utilidades básicas que un sistema operativo debe proporcionar para ser considerado compatible con Unix. Estos estándares son mantenidos por la **IEEE** (Institute of Electrical and Electronics Engineers) y la **Open Group**.
* 2)
  + Una distribución es una customización de GNU/Linux formada por una versión de kernel y determinados programas con sus configuraciones.

**Ubuntu** es amigable para principiantes, con un fuerte enfoque en la facilidad de uso.

**Fedora** se centra en la innovación, ideal para aquellos que desean lo último en tecnología.

**Arch Linux** ofrece un control total al usuario, recomendado para usuarios avanzados.

**Debian** es conocida por su estabilidad y es ideal para entornos donde la fiabilidad es crítica.

* Algunas diferencias:
* **Base:** Algunas distribuciones se basan en otras (por ejemplo, Ubuntu se basa en Debian), mientras que otras son independientes (como Arch Linux o Gentoo). La base determina en gran medida la compatibilidad del software y las herramientas disponibles.
* **Origen:** Las distribuciones pueden estar orientadas a diferentes comunidades o ser patrocinadas por corporaciones. Por ejemplo, Red Hat patrocina Fedora, mientras que Debian es un proyecto comunitario.
* **Formato de Paquetes:** Las distribuciones utilizan diferentes formatos de paquetes para instalar y actualizar software. **DEB** (usado por Debian, Ubuntu, etc.) **RPM** (usado por Fedora, Red Hat, openSUSE) **Pacman** (usado por Arch Linux)
* **Herramientas de Gestión:** Las herramientas para gestionar estos paquetes también varían. **APT** (usado por distribuciones basadas en Debian) **DNF** o **YUM** (usado por Fedora y Red Hat) **Pacman** (usado por Arch Linux)
* **Lanzamiento Regular:** Algunas distribuciones tienen ciclos de lanzamiento predecibles, con nuevas versiones cada cierto tiempo (por ejemplo, Ubuntu lanza nuevas versiones cada 6 meses).
* **Lanzamiento Rolling (Rolling Release):** Distribuciones como Arch Linux o openSUSE Tumbleweed actualizan continuamente los paquetes y no tienen "versiones" como tales. Los usuarios siempre tienen acceso a las versiones más recientes del software.
* **Soporte a Largo Plazo (LTS):** Algunas distribuciones ofrecen versiones con soporte a largo plazo, ideales para servidores y entornos de producción, como Ubuntu LTS.
* **Predeterminado:** Cada distribución suele venir con un entorno de escritorio predeterminado, como GNOME, KDE, XFCE, o LXQt. Por ejemplo, Fedora utiliza GNOME como su entorno de escritorio predeterminado, mientras que Kubuntu utiliza KDE.
* **Opciones:** Algunas distribuciones permiten seleccionar entre varios entornos de escritorio durante la instalación o a través de versiones diferentes (por ejemplo, Ubuntu tiene variantes como Xubuntu para XFCE y Lubuntu para LXQt).
* **Facilidad de Uso:** Distribuciones como Ubuntu o Linux Mint están diseñadas para ser fáciles de usar, con interfaces amigables y configuraciones predefinidas que facilitan la transición desde otros sistemas operativos.
* **Personalización y Control:** Distribuciones como Arch Linux o Gentoo ofrecen un control total al usuario sobre qué se instala y cómo se configura el sistema, pero requieren un mayor conocimiento técnico.
* **Estabilidad vs. Innovación:** Algunas distribuciones como Debian priorizan la estabilidad y son más conservadoras en la adopción de nuevas tecnologías, mientras que otras como Fedora adoptan rápidamente lo último en software y tecnologías, lo que puede implicar menor estabilidad en algunos casos.
* **Soporte de Hardware:** Algunas distribuciones están mejor equipadas para manejar una amplia gama de hardware, especialmente hardware más reciente o específico. Ubuntu, por ejemplo, tiene un excelente soporte de hardware debido a su amplio uso y comunidad.
* **Drivers y Firmware:** Algunas distribuciones, como Fedora, prefieren incluir solo software libre, lo que puede limitar el soporte para ciertos dispositivos de hardware que requieren drivers propietarios.
* **Software Libre vs. Propietario:** Algunas distribuciones como Debian se centran casi exclusivamente en software libre, mientras que otras, como Ubuntu, incluyen opciones de software propietario para facilitar la compatibilidad y la usabilidad.
* **Comunidad y Documentación:** Las distribuciones también se diferencian en el tamaño y la actividad de sus comunidades. Arch Linux, por ejemplo, es conocida por su extensa y detallada documentación, mientras que Ubuntu cuenta con una amplia comunidad de usuarios que contribuyen con soporte y recursos.
* **Software Preinstalado:** Algunas distribuciones vienen con una amplia gama de software preinstalado, como suites ofimáticas, editores de texto, y navegadores web. Otras, como Arch Linux, proporcionan sólo lo básico, permitiendo al usuario instalar solo lo que necesita.
* **Soporte Comercial:** Algunas distribuciones, como Red Hat Enterprise Linux (RHEL), ofrecen soporte comercial y suscripciones, mientras que otras, como Debian, dependen del soporte comunitario.
* **Actualizaciones de Seguridad:** La frecuencia y el método de distribución de actualizaciones de seguridad también puede variar entre distribuciones.
  + El Proyecto Debian es una asociación de personas que han hecho causa común para crear un sistema operativo (SO) libre. Este sistema operativo que hemos creado se llama Debian.

Debian lo producen cerca de un millar de desarrolladores activos, dispersos por el mundo que ayudan voluntariamente en su tiempo libre. Son pocos los desarrolladores que realmente se han encontrado en persona. La comunicación se realiza principalmente a través de correo electrónico (listas de correo en lists.debian.org) y a través de IRC (canal #debian en irc.debian.org).

Debian comenzó en agosto de 1993 gracias a Ian Murdock, como una nueva distribución que se realizaría de forma abierta, en la línea del espíritu de Linux y GNU. Debian estaba pensado para ser creada de forma cuidadosa y concienzuda, y ser mantenida y soportada con el mismo cuidado. Comenzó como un grupo de pocos y fuertemente unidos hackers de Software Libre, y gradualmente creció hasta convertirse en una comunidad grande y bien organizada de desarrolladores y usuarios

El nombre tiene su origen en los nombres del creador de Debian, Ian Murdock, y su esposa, Debra. Debian ha tenido varios líderes desde sus comienzos en el año 1993.

Debian ha tenido varios líderes desde sus comienzos en el año 1993.

Ian Murdock fundó Debian en agosto de 1993 y lo condujo hasta marzo de 1996.

Bruce Perens condujo Debian desde abril de 1996 hasta diciembre de 1997.

Ian Jackson condujo Debian desde enero de 1998 hasta diciembre de 1998.

Wichert Akkerman condujo Debian desde enero de 1999 hasta marzo de 2001.

Ben Collins condujo Debian desde abril de 2001 hasta abril de 2002.

Bdale Garbee condujo Debian desde abril de 2002 hasta abril de 2003.

Martin Michlmayr condujo Debian desde marzo de 2003 hasta marzo de 2005.

Branden Robinson condujo Debian desde abril de 2005 hasta abril de 2006.

Anthony Towns condujo Debian desde abril de 2006 hasta abril de 2007.

Sam Hocevar condujo Debian desde abril de 2007 hasta abril de 2008.

Steve McIntyre condujo Debian desde abril de 2008 hasta abril de 2010.

Stefano Zacchiroli condujo Debian desde abril de 2010 hasta abril de 2013.

Lucas Nussbaum condujo Debian desde abril de 2013 hasta abril de 2015.

Neil McGovern condujo Debian desde abril de 2015 hasta abril de 2016.

Mehdi Dogguy condujo Debian desde abril de 2016 hasta abril de 2017.

Chris Lamb condujo Debian desde abril 2017 hasta 2019.

Sam Hartman condujo Debian desde April 2019 hasta April 2020.

Jonathan Carter fue elegido April 2020 y es el actual conductor.

* 3)
  + Componentes Fundamentales:
* El kernel (núcleo)
* El Shell (interprete de comandos)
* El FileSystem (sistema de archivos)
  + La estructura básica del S.O es el Kernel, que ejecuta programas y gestiona dispositivos de hardware. Sus funciones más importantes son la administración de memoria, CPU y la E/S. El Kernel en un sentido estricto, es el sistema operativo.
* 4)
  + El **kernel** (o núcleo) es la parte central del sistema operativo. Su función principal es gestionar el hardware del sistema y permitir que el software interactúe con él. Desde su creación en 1991, ha evolucionado significativamente, incorporando mejoras en funcionalidad, soporte de hardware, y estabilidad, y continúa siendo desarrollado activamente por una comunidad global de colaboradores.
  + Funciones Más Importantes del Kernel (Administración de Recursos):

Memoria: Controla cómo se utiliza la memoria del sistema, asignando y liberando espacio según sea necesario.

CPU: Gestiona el uso del procesador, distribuyendo el tiempo de CPU entre los diferentes procesos.

E/S: Maneja las operaciones de entrada y salida, como la lectura y escritura en discos duros y otros dispositivos.

* + Hasta la fecha de la última actualización, la versión estable más reciente del kernel de Linux es la **6.4.11**, lanzada en agosto de 2023.

En las versiones anteriores a la 2.4, el esquema de versionado del kernel de Linux se basaba en un formato de número de versión que indicaba el estado de desarrollo y estabilidad del kernel.

Formato: X.Y donde X es el número principal de versión y Y es el número de versión menor.

A partir de la versión 2.6, se introdujo un esquema de versionado más detallado y sistemático para reflejar el estado de desarrollo y la madurez del kernel.

Formato: X.Y.Z donde:

X: Número de versión principal, incrementado para cambios mayores y nuevos lanzamientos importantes.

Y: Número de versión menor, incrementado para nuevas características y mejoras significativas en el kernel.

Z: Número de revisión, incrementado para correcciones de errores y mejoras menores.

* + Sí, es completamente posible tener más de un kernel de GNU/Linux instalado en la misma máquina. De hecho, es una práctica común en muchas configuraciones para proporcionar flexibilidad y seguridad.
  + Se encuentra dentro de /boot, en ese directorio se encuentran todos los kernel que posea.
  + El kernel de GNU/Linux se considera monolítico porque su diseño incluye la mayoría de las funcionalidades esenciales del sistema operativo en el mismo espacio de memoria privilegiado. Sin embargo, su capacidad para manejar módulos del núcleo de manera dinámica le otorga una característica híbrida, permitiendo flexibilidad y extensión sin necesidad de reiniciar el sistema.
* 5)
  + Un intérprete de comandos (también conocido como **shell**) es un programa que proporciona una interfaz para que los usuarios interactúen con el sistema operativo mediante comandos de texto. El shell interpreta los comandos que el usuario ingresa y los ejecuta, enviándolos al sistema operativo para que realice las operaciones correspondientes.
  + Funciones:
    - Interpretación de Comandos: El shell toma los comandos ingresados por el usuario, los interpreta y los convierte en instrucciones que el sistema operativo puede ejecutar.
    - Ejecución de Programas: Facilita la ejecución de programas y scripts. Permite al usuario iniciar y controlar programas mediante la línea de comandos.
    - Gestión de Archivos y Directorios: Proporciona comandos para crear, eliminar, copiar, mover y listar archivos y directorios, así como para gestionar permisos.
    - Automatización y Scripting: Permite la creación y ejecución de scripts, que son secuencias de comandos que automatizan tareas repetitivas.
    - Redirección y Tuberías: Soporta la redirección de entrada y salida de comandos y la conexión de varios comandos mediante tuberías para procesar datos en conjunto.
    - Variables y Control de Flujo: Ofrece la capacidad de definir y usar variables, así como estructuras de control como bucles y condicionales dentro de scripts.
  + 3 intérpretes de comando:
    - Compatibilidad y Uso:
* Bash: Amplia compatibilidad con scripts y estándares POSIX, ampliamente utilizado y bien documentado.
* Zsh: Ofrece características avanzadas y extensibilidad, pero puede ser más complejo de configurar.
* Fish: Enfocado en la facilidad de uso y características interactivas modernas, pero con una sintaxis y compatibilidad diferente.
  + - Autocompletado y Productividad:
* Bash: Buen autocompletado, pero menos avanzado en comparación con Zsh y Fish.
* Zsh: Avanzado en autocompletado y corrección ortográfica, ideal para usuarios avanzados.
* Fish: Excelente autocompletado y sugerencias en tiempo real, ideal para nuevos usuarios y quienes buscan una experiencia interactiva mejorada.
  + - Configuración y Personalización:
* Bash: Configurable con ~/.bashrc y otros archivos, pero menos extensible en comparación con Zsh y Fish.
* Zsh: Altamente configurable y extensible, especialmente con frameworks como Oh My Zsh.
* Fish: Ofrece una configuración sencilla y una interfaz gráfica, pero con menos flexibilidad en scripting.
  + En GNU/Linux, los comandos que puedes ejecutar en un shell se encuentran en ubicaciones específicas del sistema de archivos. Estos comandos pueden ser **propios** (es decir, internos al shell) o **externos** (programas y utilidades del sistema).

Los comandos internos son parte del propio intérprete de comandos y están disponibles directamente desde el shell. Estos comandos no son ejecutables externos, sino que son funcionalidades incorporadas en el shell. No tienen una ubicación específica en el sistema de archivos porque forman parte del código del propio shell. Ejemplos: cd, echo, exit, history, pwd.

Los comandos externos son programas y utilidades que están ubicados en el sistema de archivos y se ejecutan como procesos independientes. Estos comandos suelen encontrarse en directorios específicos que forman parte del PATH del sistema.

RUTA: /bin. Contiene comandos esenciales para el sistema y que son necesarios para el arranque y funcionamiento básico del sistema. Ejemplos: ls, cp, mv, cat.

* + El shell no es parte del kernel de GNU/Linux porque cumple un rol completamente diferente en el sistema operativo. El kernel gestiona los recursos del sistema y proporciona una interfaz entre el hardware y el software, mientras que el shell actúa como una interfaz de usuario para interactuar con el sistema operativo. El shell se ejecuta en el espacio de usuario y utiliza las funcionalidades proporcionadas por el kernel para llevar a cabo sus tareas.
  + Puedes definir un shell diferente para cada usuario especificándolo al crear el usuario o cambiándolo para un usuario existente.

Archivo /etc/passwd: Almacena el shell de cada usuario, y cada línea del archivo contiene el shell asignado al usuario.

Los usuarios normales pueden cambiar su propio shell, mientras que el usuario root puede cambiar el shell de cualquier usuario. Los shells deben estar listados en /etc/shells para ser válidos.

* 6)
  + Un **sistema de archivos** (o **File System** en inglés) es una estructura y método que un sistema operativo utiliza para organizar, almacenar, y gestionar los archivos en un dispositivo de almacenamiento, como un disco duro, una unidad de estado sólido (SSD), o una memoria USB. El sistema de archivos define cómo se guardan y recuperan los datos, y cómo se estructuran los directorios y archivos.
  + GNU/Linux soporta una amplia gama de sistemas de archivos, desde los tradicionales sistemas de archivos de Unix (como Ext2/3/4) hasta sistemas de archivos avanzados como Btrfs y XFS, así como sistemas de archivos de otros sistemas operativos como NTFS y exFAT.

El adoptado por GNU/Linux es el Extended (v2, v3, v4).

* + **FAT:** GNU/Linux puede visualizar y montar particiones FAT de manera nativa, utilizando herramientas estándar.

**NTFS:** GNU/Linux puede visualizar particiones NTFS y montar con soporte de lectura y escritura utilizando ntfs-3g.

* + La estructura de archivos en GNU/Linux está organizada jerárquicamente desde la raíz (/) y sigue el estándar FHS para asegurar consistencia y organización. Los directorios como /bin, /etc, /home, y /var son fundamentales para la administración y operación del sistema. El FHS (Filesystem Hierarchy Standard) proporciona una guía clara para la organización y ubicación de archivos, ayudando a mantener la coherencia entre diferentes distribuciones de GNU/Linux.
* 7)
  + Una partición es una división lógica de un disco duro físico o dispositivo de almacenamiento que actúa como una unidad independiente dentro del sistema operativo. Al particionar un disco, se crean diferentes segmentos o “partes” que el sistema puede manejar como si fueran discos separados. Cada partición puede albergar un sistema de archivos distinto, y es posible instalar diferentes sistemas operativos en particiones diferentes del mismo disco.

Tipos:

* + - Partición primaria: división cruda del disco (puede haber 4 por

disco). Se almacena información de la misma en el MBR.

* + - Partición extendida: sirve para contener unidades lógicas en su

interior. Solo puede existir una partici´on de este tipo por

disco. No se define un tipo de FS directamente sobre ella.

* + - Partición lógica: ocupa la totalidad o parte de la partición

extendida y se le define un tipo de FS. Las particiones de este

tipo se conectan como una lista enlazada.

Ventajas:

* + - Organización y Separación de Datos: Facilita la organización de los datos al permitir la separación del sistema operativo, los datos de usuario, y otros tipos de información en particiones diferentes.
    - Gestión de Múltiples Sistemas Operativos: Permite instalar y gestionar múltiples sistemas operativos en un mismo disco. Cada sistema operativo puede estar en una partición separada.
    - Mejora de la Seguridad y Recuperación: En caso de fallo del sistema operativo o corrupción de datos, las particiones separadas pueden evitar que todos los datos se vean afectados. Esto facilita la recuperación.
    - Facilita la Realización de Copias de Seguridad: Las copias de seguridad pueden ser más sencillas al centrarse en particiones específicas en lugar de todo el disco.

Desventajas:

* + - Espacio de Almacenamiento Inflexible: El espacio asignado a una partición no se puede redistribuir fácilmente. Si una partición se queda sin espacio, no se puede ampliar sin afectar a otras particiones.
    - Complejidad en la Gestión: La creación y gestión de particiones puede ser compleja, especialmente para usuarios no técnicos. Requiere un entendimiento adecuado del sistema de archivos y de las necesidades de almacenamiento.
    - Rendimiento Potencialmente Afectado: En algunos casos, tener múltiples particiones puede impactar negativamente en el rendimiento, especialmente si el disco debe leer y escribir entre diferentes particiones con frecuencia.
    - Riesgo de Pérdida de Datos: Si las particiones no se gestionan adecuadamente (por ejemplo, al redimensionar o formatear), existe el riesgo de pérdida de datos.
  + En GNU/Linux, las particiones se identifican mediante nombres de dispositivos que indican el tipo de disco y la partición específica en ese disco. Estos nombres se encuentran en el directorio /dev/ y varían según el tipo de disco (IDE, SCSI, SATA).

Los discos IDE (Integrated Drive Electronics), también conocidos como PATA (Parallel ATA), se identifican como /dev/hdxn, donde x es una letra que representa el disco en el sistema y n es el número de la partición. Ej: hda1 (Primer disco IDE, primera partición)

* + Para instalar GNU/Linux, es necesario tener al menos una partición, pero lo más recomendable es crear dos particiones principales: una para el sistema operativo (/) y otra para la memoria de intercambio (Swap).

Partición de Root (/):

* Tipo de Partición: Primaria o Lógica.
* Identificación: Generalmente /dev/sda1 (en SATA) o /dev/hda1 (en IDE).
* Tipo de File System: Ext4 es el más común, pero también pueden usarse otros como XFS, Btrfs, entre otros.
* Punto de Montaje: / (Root).

Partición de Swap:

* Tipo de Partición: Primaria o Lógica.
* Identificación: Puede ser algo como /dev/sda2 o /dev/nvme0n1p2.
* Tipo de File System: Swap (No es un sistema de archivos, es un tipo especial de partición).
* Punto de Montaje: No tiene punto de montaje como los sistemas de archivos tradicionales. Se utiliza como espacio de intercambio.
  + Algunos casos:
    - Algunos sistemas de archivos (p.e. versiones antiguas de sistemas FAT de Microsoft) tienen tamaños máximos más pequeños que los que el tamaño que proporciona un disco, siendo necesaria una partición de tamaño pequeño, para que sea posible el adecuado funcionamiento de este antiguo sistema de archivos.
    - Se puede guardar una copia de seguridad de los datos del usuario en otra partición del mismo disco, para evitar la pérdida de información importante. Esto es similar a un RAID, excepto en que está en el mismo disco.
    - En algunos sistemas operativos aconsejan más de una partición para funcionar, como, por ejemplo, la partición de intercambio (swap) en los sistemas operativos basados en Linux.
    - A menudo, dos sistemas operativos no pueden coexistir en la misma partición, o usar diferentes formatos de disco “nativo”. La unidad se particiona para diferentes sistemas operativos.
    - Uno de los principales usos que se le suele dar a las particiones (principalmente a la extendida) es la de almacenar toda la información del usuario (entiéndase música, fotos, vídeos, documentos), para que al momento de reinstalar algún sistema operativo se formatee únicamente la unidad que lo contiene sin perder el resto de la información del usuario.
  + Existen dos tipos principales de software para particionar discos en GNU/Linux:

Particionadores Destructivos:

* Descripción: Estos particionadores permiten crear y eliminar particiones, pero generalmente destruyen los datos existentes en las particiones afectadas. Son útiles cuando se desea empezar desde cero en un disco o cuando se quiere reorganizar completamente el espacio de almacenamiento. Ejemplo: fdisk.
* Ventaja: Son simples y eficaces cuando se requiere una reconfiguración completa del disco.
* Desventaja: No permiten conservar los datos existentes durante la creación o eliminación de particiones.

Particionadores No Destructivos:

* Descripción: Estos particionadores permiten crear, eliminar y modificar particiones sin destruir los datos existentes. Son ideales cuando se necesita ajustar el tamaño de una partición o crear nuevas particiones en un disco que ya contiene datos importantes. Ejemplos: fips, gparted.
* Ventaja: Ofrecen flexibilidad y seguridad al permitir modificar las particiones sin pérdida de datos.
* Desventaja: Son más complejos y lentos que los destructivos, especialmente cuando se realizan operaciones como redimensionar particiones.

Comparación:

* Uso: Los particionadores destructivos son adecuados para configuraciones iniciales o cuando se quiere reformatear un disco completo, mientras que los no destructivos son más adecuados para ajustes en sistemas ya en uso.
* Seguridad de Datos: Los particionadores no destructivos son preferibles cuando es importante preservar los datos existentes, mientras que los destructivos son más arriesgados en este sentido.
* 8)
  1. **BIOS** (Basic Input/Output System) es un firmware esencial que se encuentra integrado en la placa base de un ordenador. Se encarga de inicializar y probar los componentes hardware durante el proceso de arranque, además de cargar y ejecutar el sistema operativo.

Su función principal es la de iniciar los componentes de hardware y lanzar el sistema operativo de un ordenador cuando lo encendemos (a través del MBC). También carga las funciones de gestión de energía y temperatura del ordenador.

Cuando enciendes tu ordenador lo primero que se carga en él es el BIOS (es el primer código que se ejecuta cuando se enciende el sistema). Este firmware entonces se encarga de iniciar, configurar y comprobar que se encuentre en buen estado el hardware del ordenador, incluyendo la memoria RAM, los discos duros, la placa base o la tarjeta gráfica. Cuando termina selecciona el dispositivo de arranque (disco duro, CD, USB etcétera) y procede a iniciar el sistema operativo, y le cede a él el control de tu ordenador.

* + **UEFI (Unified Extensible Firmware Interface)** es una interfaz de firmware moderna que ha sido desarrollada por un consorcio de grandes empresas tecnológicas para actualizar y mejorar el proceso de arranque de los sistemas operativos, reemplazando al tradicional BIOS.

UEFI es una evolución significativa del BIOS, que no solo moderniza el proceso de arranque con una interfaz gráfica y soporte para sistemas de archivos más avanzados, sino que también introduce importantes mejoras en seguridad, como el Secure Boot, para proteger el sistema contra amenazas de seguridad. Además, permite una gestión más flexible y robusta del proceso de arranque, simplificando la interacción entre el firmware, el hardware y el sistema operativo.

*\*(un firmware es un software que maneja físicamente al hardware)*

* + El **Master Boot Record (MBR)** es un sector de arranque especial ubicado al comienzo de un disco duro, generalmente en el primer sector (cilindro 0, cabeza 0, sector 1). Es el primer código ejecutado al iniciar un sistema operativo en un disco de particiones MBR. Existe un MBR en todos los discos y si existiese más de un disco rígido en la máquina, sólo uno es designado como Primary Master Disk. El tamaño del MBR coincide con el tamaño estándar del sector: 512 bytes.

El **Master Boot Code (MBC)** es el código específico contenido dentro del MBR que se ejecuta en el proceso de arranque. Es el encargado de cargar y transferir el control al gestor de arranque (bootloader) ubicado en la partición activa. El MBC es un pequeño código que permite arrancar el SO. La última acción del BIOS es leer el MBC, lo lleva a memoria y lo ejecuta.

* + Las siglas **GPT** se refieren a **GUID Partition Table** (Tabla de Particiones GUID). Es un estándar para el esquema de particionamiento de discos que forma parte de la especificación UEFI (Unified Extensible Firmware Interface). GPT fue creado para superar las limitaciones del esquema tradicional de particiones MBR (Master Boot Record). GPT especifica la ubicación y formato de la tabla de particiones en un disco duro, es parte de EFI y puede verse como una sustitución del MBR.
  + Un gestor de arranque (en inglés «bootloader») es un programa sencillo que no tiene la totalidad de las funcionalidades de un sistema operativo, y que está diseñado exclusivamente para preparar todo lo que necesita el sistema operativo para funcionar. Normalmente se utilizan los cargadores de arranque multietapas, en los que varios programas pequeños se suman los unos a los otros, hasta que el último de ellos carga el sistema operativo.

En los ordenadores modernos, el proceso de arranque comienza cuando la unidad central de procesamiento ejecuta los programas contenidos en una memoria de sólo lectura en una dirección predefinida y se configura la unidad central para ejecutar este programa, sin ayuda externa, al encender el ordenador.

Habitualmente se instala en el MBR y asume el rol de MBC. En una computadora en la que hay sólo un sistema operativo, no hay referencias a pantalla generalmente. Si hay un gestor de arranque, este programa nos permitirá elegir el sistema operativo a arrancar. El código del MBC de Windows, por ejemplo, busca en la tabla de particiones cual es la primera partición primaria con el flag de “booteo” activo y transfiere el control al código que se encuentra al comienzo de dicha partición: el PBR (partition boot record).

En el caso del sistema operativo Linux, se puede optar por distintos gestores de arranque, por ejemplo, LILO (Linux Loader), GRUB (Grand Unified Bootloader) o GAG (Gestor de arranque Gráfico). LILO no se basa en un sistema de archivos específico. Funciona en una variedad de sistemas de archivos. GRUB en cambio, debe comprender el sistema de archivos y el formato de los directorios. GRUB tiene algunas ventajas con respecto a LILO: tiene una línea de comandos interactiva como, permite arrancar desde una red, y podría considerarse más seguro.

* + El BIOS (Basic I/O System) es el responsable de iniciar la carga del SO a través del MBC. Este carga el programa de booteo (desde el MBR). El gestor de arranque lanzado desde el MBC carga el Kernel: prueba y hace disponibles los dispositivos y luego pasa el control al proceso init.

El proceso de arranque se ve como una serie de pequeños programas de ejecución encadenada.

Pasos resumidos:

* POST: Verificación inicial del hardware.
* Carga del Firmware/BIOS o UEFI: Configuración del hardware y selección del dispositivo de arranque.
* Localización y Carga del Gestor de Arranque: Carga del gestor de arranque desde el MBR o la partición EFI.
* Ejecución del Gestor de Arranque: Selección del sistema operativo y carga del núcleo.
* Transferencia de Control al Núcleo: Inicialización del núcleo y montaje del sistema de archivos raíz.
* Ejecución del Proceso de Inicio: Inicio de servicios y presentación del entorno de usuario.
  + Proceso de Arranque en GNU/Linux:
    - **POST:** Verificación del hardware por el BIOS.
    - **Carga del MBR:** El BIOS lee el MBR del disco, cargando el Master Boot Code.
    - **Ejecución de LILO:** LILO (o GRUB) se encarga de cargar el núcleo.
    - **Carga del Núcleo:** El kernel de GNU/Linux es cargado en memoria y configurado.
    - **Ejecución de Init:** init arranca el resto de los procesos y servicios del sistema.
    - **Interfaz de Usuario:** El sistema presenta la interfaz de usuario para interacción.
  + Proceso de ShutDown en GNU/Linux:
    - **Comando de Apagado**: El proceso de apagado generalmente se inicia mediante un comando como shutdown, poweroff, o halt, que puede ser ejecutado desde una terminal o a través de una interfaz gráfica.
    - **Notificación a Servicios**: El sistema notifica a todos los servicios y daemons en ejecución que se está iniciando el proceso de apagado. Esto permite que los servicios puedan cerrar sus conexiones y liberar recursos de manera ordenada.
    - **Notificación a Usuarios**: Se envían mensajes de advertencia a todos los usuarios conectados para informarles sobre el inminente apagado. Esto puede incluir una notificación para guardar el trabajo y cerrar las sesiones.
    - **Detención de Procesos**: El sistema detiene todos los procesos en ejecución de forma ordenada. Esto incluye cerrar aplicaciones de usuario, servicios del sistema y procesos en segundo plano.
    - **Desmontaje de Sistemas de Archivos**: Se desmontan todos los sistemas de archivos montados para asegurarse de que no haya datos en uso o archivos abiertos. Esto previene la pérdida de datos y la corrupción del sistema de archivos.
    - **Sincronización de Datos**: El sistema sincroniza todos los datos pendientes en disco. Esto garantiza que todos los datos en caché sean escritos en el disco y que no haya pérdida de datos.
    - **Apagado del Hardware**: Finalmente, el sistema apaga el hardware. Dependiendo del comando utilizado y de la configuración del sistema, esto puede implicar el apagado completo de la máquina o simplemente la suspensión.
  + Sí, es posible tener en una PC GNU/Linux junto con otro sistema operativo instalado. Este escenario se conoce como **arranque dual (dual-boot)** o **multiarranque (multi-boot)**. Ya que un disco rígido puede particionarse y en cada partición tener un sistema de archivos distinto (partición primaria), sería como tener varios discos distintos, uno con cada SO, por lo tanto, se necesitará un gestor de arranque como los descriptos arriba.
* 9)
  + La base del sistema de archivos de Linux, es obviamente el archivo, que no es otra cosa que la estructura empleada por el sistema operativo para almacenar información en un dispositivo físico como un disco duro, un disquete, un CD-ROM o un DVD. Como es natural un archivo puede contener cualquier tipo de información, desde una imagen en formato PNG o JPEG a un texto o una página WEB en formato HTML.

El sistema de archivos es la estructura que permite que Linux maneje los archivos que contiene. Todos los archivos de Linux tienen un nombre, el cual debe cumplir unas ciertas reglas:

* Un nombre de archivo puede tener entre 1 y 255 caracteres.
* Se puede utilizar cualquier carácter excepto la barra inclinada / y no es recomendable emplear los caracteres con significado especial en Linux, que son los siguientes: = ^ ~ ' " ` \* ; - ? [ ] ( ) ! & ~ < >. Para emplear ficheros con estos caracteres o espacios hay que introducir el nombre del fichero entre comillas.
* Se pueden utilizar números exclusivamente si así se desea.
* Las letras mayúsculas y minúsculas se consideran diferentes, y por lo tanto no es lo mismo carta.txt que Carta.txt ó carta.Txt.

Como en Windows, se puede emplear un cierto criterio de "tipo" para marcar las distintas clases de ficheros empleando una serie de caracteres al final del nombre que indiquen el tipo de fichero del que se trata. Así, los ficheros de texto, HTML, las imágenes PNG o JPEG tienen extensiones .txt, .htm (o .html), .png y .jpg (o .jpeg) respectivamente. Pese a esto Linux sólo distingue tres tipos de archivos:

* Archivos o ficheros ordinarios, son los mencionados anteriormente.
* Directorios (o carpetas), es un archivo especial que agrupa otros ficheros de una forma estructurada.
* Archivos especiales, son la base sobre la que se asienta Linux, puesto que representan los dispositivos conectados a un ordenador, como puede ser una impresora. De esta forma introducir información en ese archivo equivale a enviar información a la impresora. Para el usuario estos dispositivos tienen el mismo aspecto y uso que los archivos ordinarios.
  + Funcionamiento de:

**Editor vi:** vi es un editor de texto clásico en sistemas Unix y GNU/Linux. Es conocido por su modo de operación basado en comandos y su capacidad de ser utilizado en terminales sin interfaz gráfica.

Modos en vi:

* Modo de Comando: Es el modo predeterminado. Aquí puedes moverte por el texto, borrar, copiar, pegar, y realizar otras operaciones de edición.
* Modo de Inserción: En este modo, puedes insertar texto. Se accede a este modo presionando i desde el modo de comando.
* Modo de Línea de Comando: Permite ejecutar comandos para guardar, salir y realizar otras acciones. Se accede a este modo presionando : desde el modo de comando.

Comandos Básicos:

* Abrir un archivo: vi archivo.txt
* Entrar en modo de inserción: Presiona i (inserta texto antes del cursor) o a (añade texto después del cursor).
* Guardar y salir: En modo de comando, escribe :wq y presiona Enter.
* Salir sin guardar: En modo de comando, escribe :q! y presiona Enter.
* Mover el cursor: Usa las teclas de flecha h, j, k, l(izquierda, abajo, arriba, derecha).

**Editor mcedit:** Mcedit es un editor de texto basado en el programa Midnight Commander (MC), que es un administrador de archivos en modo texto. Mcedit es más amigable y tiene una interfaz de usuario basada en menús y una apariencia más gráfica comparado con vi.

Características:

* + Interfaz de Usuario: Proporciona una interfaz de edición de texto más fácil de usar con menús y atajos de teclado.
  + Operaciones Básicas: Permite editar texto, buscar y reemplazar texto, copiar y pegar bloques de texto, entre otras funcionalidades.

Comandos Básicos:

* + Abrir un archivo: mcedit archivo.txt
  + Guardar archivo: Presiona F2 (guardar) en la interfaz.
  + Salir del editor: Presiona F10 para salir.

**Comando cat :** El comando cat (concatenate) se utiliza para mostrar el contenido de uno o más archivos en la salida estándar (generalmente la terminal).

Uso Básico:

* + Mostrar el contenido de un archivo: cat archivo.txt
  + Concatenar y mostrar el contenido de varios archivos: cat archivo1.txt archivo2.txt
  + Crear un nuevo archivo (redirigiendo la entrada): cat > nuevoarchivo.txt (escribe el texto y presiona Ctrl+D para guardar y salir).

**Comando more :** El comando more se utiliza para ver el contenido de archivos de texto paginados. Permite desplazarse por el contenido de un archivo página por página, lo que es útil para archivos grandes.

Uso Básico:

* + Mostrar el contenido de un archivo de forma paginada: more archivo.txt
  + Navegar por el archivo:
    - Presiona Space para avanzar una página.
    - Presiona Enter para avanzar una línea.
    - Presiona q para salir de more.
  + Pasos:
* vi prueba.exe
* 21016/6 camila tarrio
* esc
* :wq + enter
  + Cuando ejecutas file sobre un archivo, este lo examina y devuelve información sobre su tipo.

Ej: file prueba.exe ---> ejemplo.txt: ASCII text

* 10)
  1. El comando mkdir(make directory) se utiliza para crear directorios o carpetas en el sistema de archivos de Linux. Cuando ejecutas este comando, crea un nuevo directorio con el nombre especificado, en la ubicación actual o en otra ruta que definas.

→ mkdir ISO2017

* 1. El comando cd se usa para cambiar el directorio actual a otro especificado.

→ cd ISO2017

* 1. El comando touch se usa para crear archivos vacíos o actualizar la marca de tiempo (timestamp) de archivos existentes. Si el archivo no existe, touch lo creará. Si el archivo ya existe, simplemente actualizará su fecha de acceso y modificación sin alterar su contenido.

→ touch iso2017-1 iso2017-2

* 1. El comando ls (list) se usa para mostrar una lista de archivos y directorios en el directorio actual o en un directorio específico. Si no se especifica ninguna ruta, ls muestra el contenido del directorio en el que te encuentras actualmente.

→ ls ---> (muestra iso2017-1 iso2017-2)

* 1. El comando pwd (print working directory) muestra la ruta completa (absoluta) del directorio actual. Esto es útil para saber exactamente en qué ubicación del sistema de archivos te encuentras en un momento dado.

→ pwd ---> (muestra /home/camila/ISO2017)

* 1. El comando find busca archivos y directorios dentro de un directorio especificado (o en el directorio actual si no se especifica otro) que coincidan con ciertos criterios. En este caso, -name se usa para buscar archivos por su nombre.
* **.**: Indica el directorio actual como punto de partida para la búsqueda.
* **-name**: Indica que se buscarán archivos cuyo nombre coincida con el patrón especificado.
* **"iso\*"**: Es el patrón que estamos buscando, donde \* significa cualquier cantidad de caracteres después de "iso" (puede ser vacío o cualquier texto).

→ find . -name “iso\*” ---> (muestra ./iso2017-1 y ./iso2017-2)

* 1. Para informar la cantidad de espacio libre en disco en Linux, se utiliza el comando df. Además, si agregamos el parámetro -h (human-readable) convierte las unidades de bytes en un formato más comprensible, cómo KB, MB, o GB.

→ df [muestra por ej: (s. ficheros) udev (bloques de 1k) 1466820 (usados) 0…]

→ df -h [muestra por ej: (s. ficheros) udev (tamaño) 1.4G (usados) 0…]

* 1. Para verificar los usuarios conectados actualmente al sistema en Linux, se utiliza el comando who. Este comando muestra una lista de usuarios conectados al sistema, indicando también la terminal que están usando, la hora de inicio de sesión y, en algunos casos, desde qué dirección IP o máquina están conectados.

→ who (muestra camila tty1 2024-09-22 15:37)

* 1. Para acceder al archivo **iso2017-1** y editarlo para ingresar tu nombre y apellido, puedes utilizar un editor de texto en la terminal, como nano o vi.

→ vi iso2017-1

→ i

→ Camila Tarrio

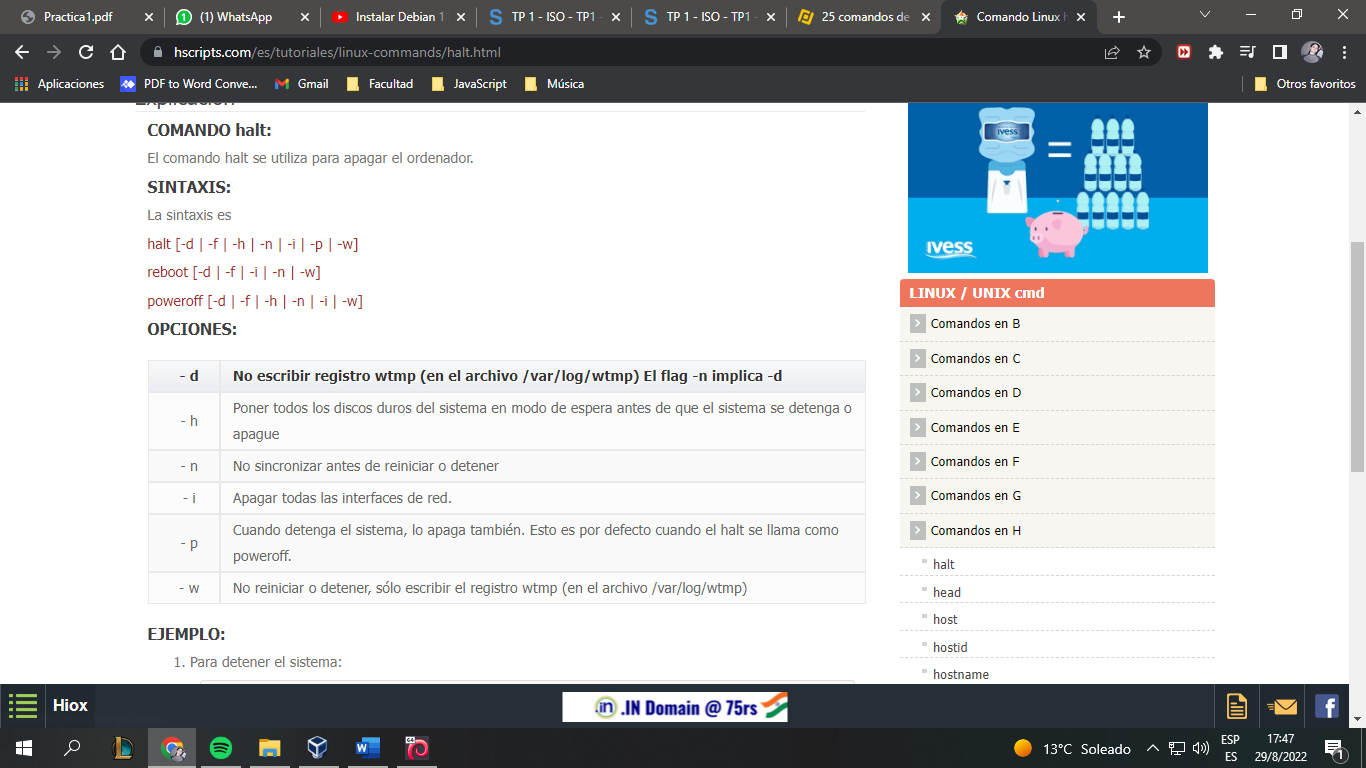
→ esc + :wq + enter

* 1. Para mostrar las últimas líneas de un archivo en Linux, se utiliza el comando tail. Por defecto, tail muestra las últimas 10 líneas del archivo especificado.

→ tail 2017-1 (muestra Camila Tarrio)

* 11)
  + El comando shutdown permite a los usuarios con privilegios suficientes (normalmente, el usuario root) apagar, reiniciar o poner en modo de espera el sistema. El uso adecuado de shutdown ayuda a cerrar todos los procesos y servicios de manera segura antes de apagar el sistema.
* shutdown [Opciones] [Tiempo] [Mensaje]
* **tiempo**: Puede ser un número que representa la cantidad de minutos hasta que se realice el apagado, o puede ser una hora específica en formato HH
* **mensaje**: Opcionalmente, puedes enviar un mensaje a todos los usuarios conectados para informarles del apagado.
* **-h**: Apaga el sistema de forma segura. Esta opción es la predeterminada si no se especifica otra acción.
* **-r**: Reinicia el sistema en lugar de apagarlo.
* **-c**: Cancela un apagado programado. Esto es útil si ya se ha configurado un apagado y decides no proceder.
* **-k**: Envía un mensaje a los usuarios conectados, pero no apaga el sistema. Utiliza esta opción si solo deseas advertir a los usuarios sin realizar un apagado.
* **-f**: Forzar el apagado. Esto omite los procesos de cierre y apaga el sistema de inmediato, lo cual no es recomendable a menos que sea necesario.
* **-P**: Apaga el sistema (similar a -h), pero se usa para especificar el comportamiento de algunos sistemas que requieren esto para un apagado limpio.
* Ej: shutdown -h +10 "El sistema se apagará en 10 minutos. Por favor, guarden su trabajo."
  + El comando reboot reinicia el sistema de manera inmediata. Al ejecutarlo, el sistema cierra todos los procesos y servicios en ejecución, y luego se reinicia. Es una forma efectiva de reiniciar el sistema sin necesidad de apagarlo primero.
* reboot [Opciones]
* **-f**: Forzar el reinicio. Omite el cierre limpio de los procesos y reinicia el sistema inmediatamente. Utilizar con precaución, ya que puede causar pérdida de datos.
* Si deseas reiniciar el sistema en un tiempo específico, es mejor usar shutdown con la opción -r.
  + El comando halt detiene el sistema sin apagar físicamente la máquina. Esencialmente, el sistema se lleva a un estado de inactividad, pero la alimentación eléctrica del hardware no se apaga. Este comando puede ser utilizado para sistemas que requieren ser apagados manualmente o para realizar mantenimiento.
* halt [Opciones]
* **-f**: Forzar la detención del sistema sin realizar un cierre limpio de los procesos. Esto puede resultar en pérdida de datos y no se recomienda a menos que sea absolutamente necesario.

Tabla:



* + El comando locate en Linux se utiliza para encontrar rápidamente archivos y directorios en el sistema de archivos. Funciona consultando una base de datos que almacena la ubicación de archivos, lo que lo hace más rápido que otros métodos de búsqueda.
* locate [Opciones] patrón
* **-i**: Realiza una búsqueda insensible a mayúsculas y minúsculas.
* **-c**: Muestra solo el conteo de coincidencias en lugar de listar los archivos encontrados.
* **-r**: Permite usar expresiones regulares para la búsqueda.
  + El comando uname en Linux se utiliza para mostrar información sobre el sistema operativo y el hardware. Proporciona detalles útiles sobre el entorno en el que se está ejecutando el comando. Uname puede mostrar información sobre el sistema operativo, el nombre de la máquina, la versión del kernel y más. Sin opciones, el comando por defecto muestra el nombre del kernel.
* uname [Opciones]
* -a: Muestra un resumen completo de la información del sistema de un solo vistazo.
* **-s**: Muestra el nombre del kernel.
* **-n**: Muestra el nombre del nodo (hostname) de la máquina.
* **-r**: Muestra la versión del kernel.
* **-v**: Muestra la fecha de compilación del kernel.
* **-m**: Muestra la arquitectura del hardware (por ejemplo, x86\_64, arm).
* **-p**: Muestra la arquitectura del procesador (puede no estar disponible en todas las versiones).
* **-o**: Muestra el nombre del sistema operativo.
  + El comando dmesg en Linux se utiliza para imprimir o controlar el buffer de mensajes del núcleo (kernel). Este buffer contiene mensajes relacionados con el sistema, como mensajes de inicio del sistema, errores de hardware y otros eventos importantes.
* dmesg [Opciones]
* **-c**: Limpia el buffer de mensajes después de mostrarlos.
* **-n**: Cambia el nivel de prioridad de los mensajes que se mostrarán (por ejemplo, para mostrar solo mensajes de nivel de error o más críticos).
* **-f**: Filtra mensajes por tipo de mensaje.
* **-h**: Muestra la ayuda sobre el uso del comando y sus opciones.
* **-s**: Especifica el tamaño del buffer de mensajes en bytes.
* **-w**: Muestra los mensajes del kernel en tiempo real. La opción -w (o --follow) permite que el comando continúe ejecutándose y muestre nuevos mensajes a medida que se generan.
  + El comando lspci en Linux se utiliza para mostrar información sobre los dispositivos conectados al bus PCI (Peripheral Component Interconnect) del sistema. Esto incluye tarjetas de red, adaptadores gráficos, controladores de almacenamiento y otros periféricos conectados. Lspci escanea el bus PCI y muestra una lista de dispositivos conectados. Proporciona información detallada sobre cada dispositivo, como su identificador, clase, proveedor y más.
* lspci [Opciones]
* **-v**: Muestra información detallada sobre cada dispositivo (verbose).
* **-vv**: Muestra información aún más detallada.
* **-nn**: Muestra los identificadores del proveedor y del dispositivo en formato numérico.
* **-k**: Muestra los controladores utilizados por cada dispositivo.
* **-s**: Filtra la salida para mostrar solo el dispositivo con el identificador especificado.
* **-t**: Muestra la jerarquía de dispositivos en un formato de árbol.
  + El comando at en Linux se utiliza para programar tareas que se ejecutarán una sola vez en un momento específico en el futuro. Es una herramienta útil para automatizar la ejecución de scripts o comandos sin necesidad de intervención manual.
* at [Opciones] [Hora]
* **-f**: Especifica un archivo que contiene los comandos a ejecutar.
* **-q**: Especifica la cola a la que se añadirá el trabajo.
* Ej: at -f archivo.txt 15:30
  + El comando netstat (abreviatura de "network statistics") en Linux se utiliza para mostrar información sobre las conexiones de red, las tablas de enrutamiento, las interfaces de red y otras estadísticas relacionadas con la red. Es una herramienta muy útil para diagnosticar problemas de red y obtener información sobre el estado de las conexiones. Netstat permite visualizar detalles sobre las conexiones de red activas y los puertos en uso, así como información sobre el tráfico de red.
* netstat [Opciones]
* **-a**: Muestra todas las conexiones y puertos de escucha.
* **-n**: Muestra direcciones y números de puerto en formato numérico (sin resolver nombres).
* **-l**: Muestra solo los puertos de escucha.
* **-s**: Muestra estadísticas detalladas sobre protocolos (TCP, UDP, ICMP, etc.).
  + El comando mount en Linux se utiliza para montar sistemas de archivos y hacer que estén accesibles desde el sistema operativo. Este comando es fundamental para trabajar con dispositivos de almacenamiento como discos duros, unidades USB y particiones.
* mount [Opciones] [Dispositivo] [Punto de Montaje]
* **-t**: Especifica el tipo de sistema de archivos (ej. ext4, vfat, ntfs).
* **-o**: Permite especificar opciones de montaje, como: ro,rw,noexec,users.
* Ej: mount -t ext4 /dev/sda1 /mnt/data
  + El comando umount en Linux se utiliza para desmontar sistemas de archivos que han sido montados previamente. Desmontar un sistema de archivos es un paso importante para garantizar que todos los datos se hayan escrito correctamente y que el dispositivo se pueda desconectar de manera segura.
* umount [Opciones] [Punto de Montaje | Dispositivo]
* **-l**: Desmontar el sistema de archivos de manera "perezosa". Esto significa que el sistema de archivos se marcará como desmontado, pero el desmontaje real se realizará cuando ya no haya procesos usando el sistema de archivos.
* **-f**: Forzar el desmontaje. Útil en situaciones donde el sistema de archivos no se puede desmontar normalmente, aunque puede resultar en pérdida de datos.
  + El comando head en Linux se utiliza para mostrar las primeras líneas de uno o más archivos de texto. Es útil para obtener una vista rápida del contenido de un archivo sin necesidad de abrirlo completamente.
* head [Opciones] [Archivo]
* **-n [número]**: Especifica cuántas líneas mostrar. Si se omite, muestra las primeras 10 líneas.
* **-q**: Suprime la impresión del encabezado de nombre de archivo al mostrar múltiples archivos.
* **-v**: Muestra el nombre del archivo antes de las líneas.
  + El comando losetup en Linux se utiliza para gestionar dispositivos de bucle (loop devices). Los dispositivos de bucle permiten a los usuarios montar archivos de imagen como si fueran dispositivos de bloque reales, lo que es útil para trabajar con imágenes de disco, archivos ISO y otros tipos de archivos que representan sistemas de archivos.
* losetup [Opciones] [Dispositivo] [Archivo]
* **-a**: Muestra todos los dispositivos de bucle y los archivos que están montados en ellos.
* **-d**: Desasigna un dispositivo de bucle.
* **--find**: Encuentra el primer dispositivo de bucle disponible.
* **--show**: Muestra el nombre del dispositivo de bucle asignado.
  + El comando write en Linux se utiliza para enviar mensajes a otros usuarios que están conectados al mismo sistema. Permite la comunicación en tiempo real entre usuarios, lo que puede ser útil en entornos colaborativos o para enviar notificaciones rápidas.
* write [Usuario] [Tty]
* **usuario**: El nombre del usuario al que deseas enviar un mensaje.
* **tty**: (Opcional) El terminal específico al que deseas enviar el mensaje. Si no se especifica, se utilizará el terminal actual del usuario.
  + El comando mkfs en Linux se utiliza para crear un sistema de archivos en un dispositivo de almacenamiento. Este comando es esencial para preparar una partición o un dispositivo para su uso, permitiendo que el sistema operativo pueda leer y escribir datos en él.
* mkfs.[Tipo] [Opciones] [Dispositivo]
* **[tipo]**: Tipo de sistema de archivos que deseas crear (por ejemplo, ext4, vfat, ntfs, etc.).
* **[dispositivo]**: La partición o el dispositivo donde se creará el sistema de archivos (por ejemplo, /dev/sdb1).
* **-L [etiqueta]**: Asigna una etiqueta al sistema de archivos.
* **-m [porcentaje]**: Especifica el porcentaje de bloques reservados para el usuario root.
* **-n**: No formatear el dispositivo, solo mostrar qué haría mkfs.
  + El comando fdisk en Linux se utiliza para gestionar particiones en discos duros y otros dispositivos de almacenamiento. Proporciona una interfaz de línea de comandos para crear, eliminar y modificar particiones, así como para visualizar la tabla de particiones de un dispositivo.
* fdisk [Opciones] [Dispositivo]
* **-l**: Lista todas las particiones de todos los discos.
* **-u**: Muestra el tamaño de las particiones en sectores en lugar de cilindros (en algunas versiones).
* 12)
  1. /bin: Aquí están los comandos que pueden usar todos los usuarios (incluido el root).