

Introducción a los Sistemas Operativos

Práctica 1

**Objetivo**

El objetivo de esta práctica es que el alumno se familiarice con los conceptos básicos del sistema operativo *GNU/Linux*, así como con su entorno y comandos principales.

1. Características de *GNU/Linux*:
2. Mencione y explique las características más relevantes de *GNU/Linux*.
3. Mencione otros sistemas operativos y compárelos con *GNU/Linux* en cuanto a los puntos mencionados en el inciso *a*.
4. ¿Qué es **GNU**?
5. 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.
6. ¿Qué es **POSIX**?

A) Mencione y explique las características más relevantes de *GNU/Linux.*

**Código abierto**: GNU/Linux es de código abierto, lo que significa que cualquier usuario puede ver, modificar y distribuir el código fuente según lo considere necesario.

**Portabilidad**: Puede ser instalado en varios tipos de hardware, desde computadoras personales hasta servidores, smartphones, tablets, y otros dispositivos.

**Seguridad**: Diseñado con características de seguridad sólidas, como la separación de usuarios y el sistema de permisos.

**Multitarea**: Puede manejar múltiples tareas al mismo tiempo.

**Multiusuario**: Varios usuarios pueden usar el sistema al mismo tiempo sin interferir entre ellos.

**Compatibilidad con POSIX**: Cumple con los estándares POSIX, lo que facilita la portabilidad de aplicaciones entre sistemas operativos compatibles.

**Shell**: Proporciona una interfaz de línea de comandos poderosa y versátil.

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

**Windows**: Propietario (no es de código abierto); menos portabilidad en comparación con Linux; interfaz gráfica centrada; generalmente considerado menos seguro que Linux.

**MacOS**: Basado en UNIX y comparte algunas similitudes con Linux; propietario (no es de código abierto); específico para hardware Apple; generalmente seguro debido a su base UNIX.

**BSD**: Como Linux, es una derivación de UNIX y tiene muchas similitudes; código abierto; licencia más permisiva que la GPL de GNU.

**Solaris**: Basado en UNIX; fue propietario, pero después tuvo una versión de código abierto; utilizado principalmente en entornos empresariales.

C) ¿Qué es **GNU**?

**GNU** significa "GNU's Not Unix". Es un proyecto iniciado por Richard Stallman en 1983 para desarrollar un sistema operativo completamente libre y de código abierto que sea similar a Unix, pero que no contenga ningún código de Unix.

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.

El Proyecto **GNU** fue lanzado en 1983 por Richard Stallman con el objetivo de crear un sistema operativo completamente libre. La principal motivación de Stallman era el deseo de un entorno de computación libre y abierta. El proyecto empezó con el desarrollo de herramientas básicas como compiladores y editores (por ejemplo, GCC y Emacs). Hacia finales de la década de 1980, el proyecto GNU había producido o adquirido la mayoría de los componentes esenciales de un sistema operativo libre, excepto el núcleo. El núcleo, Linux, fue desarrollado por Linus Torvalds en 1991 y, cuando se combinó con las herramientas del proyecto GNU, se formó el sistema operativo GNU/Linux.

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

La multitarea es la capacidad de un sistema operativo para administrar y ejecutar simultáneamente múltiples tareas o procesos. Esto permite a los usuarios ejecutar múltiples aplicaciones al mismo tiempo. GNU/Linux, al igual que muchos sistemas operativos modernos, utiliza la multitarea, permitiendo la ejecución concurrente de múltiples procesos y aplicaciones.

F) ¿Qué es POSIX?

POSIX (Interfaz de Sistema Operativo Portátil) es un conjunto de estándares definidos por el IEEE para mantener la compatibilidad entre sistemas operativos. POSIX define la interfaz de aplicación, junto con las interfaces de shell y utilidades, para la compatibilidad del software con variantes de Unix y otros sistemas operativos. GNU/Linux cumple con los estándares POSIX, lo que facilita la portabilidad de aplicaciones entre sistemas operativos compatibles.

1. Distribuciones de *GNU/Linux*:
   1. ¿Qué es una distribución de *GNU/Linux*? Nombre al menos 4 distribuciones de *GNU/Linux* y cite diferencias básicas entre ellas.
   2. ¿En qué se diferencia una distribución de otra?
   3. ¿Qué es **Debian**? Acceda al sitio [[1]](#footnote-1) e indique cuáles son los objetivos del proyecto y una breve cronología del mismo

A) ¿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 es una colección o paquete que contiene el núcleo Linux junto con un conjunto de aplicaciones, herramientas, bibliotecas y servicios. Cada distribución se empaqueta y mantiene generalmente por una organización o individuo específico y puede tener objetivos, audiencias y filosofías particulares.

Ubuntu: Una de las distribuciones más populares para usuarios de escritorio y servidores. Se basa en Debian, pero tiene su propio entorno de escritorio (anteriormente Unity, ahora GNOME) y ciclos de lanzamiento. Es conocido por su facilidad de uso y amplia comunidad.

Fedora: Patrocinada por Red Hat, es una distribución que presenta las últimas tecnologías y software. Se utiliza tanto en entornos de escritorio como de servidor.

Arch Linux: Orientado a usuarios experimentados, ofrece un sistema de "rolling release" donde el software se actualiza constantemente. El usuario tiene mucho control sobre la personalización.

openSUSE: Tiene dos principales ediciones: Tumbleweed (rolling release) y Leap (versiones estables). Es conocida por su herramienta de configuración YaST.

B) ¿En qué se diferencia una distribución de otra?

Las distribuciones pueden diferir en varios aspectos:

Base de software: Algunas distribuciones pueden basarse en otras (por ejemplo, Ubuntu se basa en Debian).

Ciclo de lanzamiento: Mientras que algunas distribuciones tienen lanzamientos regulares y estables (como Ubuntu), otras son "rolling release", lo que significa que el software se actualiza constantemente (como Arch Linux).

Gestión de paquetes: Las herramientas para instalar y gestionar software pueden variar (por ejemplo, apt en Debian/Ubuntu, pacman en Arch Linux, dnf en Fedora).

Entorno de escritorio: GNOME, KDE, XFCE, entre otros.

Filosofía: Por ejemplo, Debian se adhiere a una estricta filosofía de software libre, mientras que otras distribuciones pueden incluir software propietario.

Público objetivo: Algunas distribuciones están diseñadas para usuarios principiantes, mientras que otras están destinadas a usuarios más avanzados o técnicos.

C) ¿Qué es Debian?

Debian es una distribución de GNU/Linux que se adhiere a una filosofía de software libre. Es conocida por su estabilidad, su extenso repositorio de paquetes y su sistema de gestión de paquetes "apt". Muchas otras distribuciones, como Ubuntu, se basan en Debian.

Objetivos del proyecto Debian:

Producir un sistema operativo completamente libre.

Compromiso con la calidad y la estabilidad.

Comunidad abierta y transparente.

Cronología (breve) de Debian:

1993: Ian Murdock funda el Proyecto Debian.

1996: Se adopta el sistema de gestión de paquetes "dpkg".

1997: Aparece apt, una interfaz para dpkg que simplifica la gestión de paquetes y las dependencias.

2005: Lanzamiento de Debian 3.1 (nombre en código "Sarge"), que tuvo un ciclo de desarrollo particularmente largo.

2019: Lanzamiento de Debian 10 (nombre en código "Buster").

1. Estructura de *GNU/Linux*:
   1. Nombre cuales son los 3 componentes fundamentales de *GNU/Linux*.
   2. Mencione y explique la estructura básica del Sistema Operativo *GNU/Linux*.

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

El sistema Linux está formado por tres cuerpos principales de código, en línea con la mayoría de las implementaciones UNIX más tradicionales:

1-Kernel: El Kernel es responsable de mantener todas las abstracciones importantes del sistema operativo, incluyendo elementos tales como la memoria virtual y los procesos.

2-Bibliotecas del sistema: Las bibliotecas del sistema definen un conjunto estándar de funciones mediante las que las aplicaciones pueden interactuar con el kernel. Estas funciones implementan buena parte de la funcionalidad del sistema operativo que no necesita los privilegios completos del código del kernel.

3-Utilidades del sistema: Las utilidades del sistema son programas que realizan tareas individuales y especializadas de gestión. Algunas utilidades del sistema pueden ser invocadas para inicializar y configurar algunos aspectos del sistema; otras pueden ejecutarse de manera permanente, gestionando tareas tales como responder a las conexiones entrantes de red, aceptar solicitudes de inicio de sesión por parte de los terminales y actualizar los archivos de registro.

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

La estructura básica del sistema operativo GNU/Linux se organiza en capas y componentes:

Núcleo (Kernel): En la capa más baja se encuentra el núcleo del sistema, que es el kernel Linux en el caso de GNU/Linux. El kernel controla el acceso y la gestión del hardware, así como la programación y ejecución de procesos.

Sistema GNU: Encima del kernel, se encuentra el sistema GNU, que proporciona utilidades y bibliotecas esenciales para el funcionamiento del sistema. Estas herramientas incluyen comandos de línea de comandos, bibliotecas compartidas, controladores de dispositivos y utilidades de sistema.

Entorno de usuario: La capa superior es el entorno de usuario, que puede ser una interfaz de línea de comandos (CLI) o una interfaz gráfica de usuario (GUI). En el caso de una GUI, se utiliza un entorno de escritorio que proporciona ventanas, iconos, menús y otras características visuales para que los usuarios interactúen con el sistema de manera más intuitiva.

Aplicaciones de usuario: Sobre la capa del entorno de usuario, se ejecutan las aplicaciones y programas específicos que los usuarios utilizan para realizar tareas, como navegadores web, editores de texto, reproductores multimedia y mucho más.

En resumen, el núcleo Linux se encuentra en el nivel más bajo y maneja la interacción con el hardware, el sistema GNU proporciona las herramientas y bibliotecas esenciales para el sistema, y el entorno de usuario ofrece la interfaz a través de la cual los usuarios interactúan con el sistema y ejecutan aplicaciones. Las aplicaciones de usuario se ejecutan en la capa superior, construyendo una estructura completa y funcional de sistema operativo GNU/Linux.

1. *Kernel*:
   1. ¿Qué es? Indique una breve reseña histórica acerca de la evolución del Kernel de *GNU/Linux*.
   2. ¿Cuáles son sus funciones principales?
   3. ¿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?
   4. ¿Es posible tener más de un Kernel de *GNU/Linux* instalado en la misma máquina?
   5. ¿Dónde se encuentra ubicado dentro del File System?
   6. ¿El Kernel de *GNU/Linux* es monolítico? Justifique.

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

El kernel es el núcleo del sistema operativo. Es responsable de gestionar los recursos del sistema, como la memoria, los dispositivos y el tiempo de CPU. El kernel actúa como intermediario entre las aplicaciones y el hardware del sistema.

Reseña histórica del Kernel de GNU/Linux: 1991: Linus Torvalds inicia el desarrollo del kernel Linux como un proyecto personal. Se lanza la versión 0.01.

1992: Se adopta la Licencia GPL para el kernel.

1994: Se lanza Linux 1.0.

1996: Lanzamiento de Linux 2.0, que incluye soporte para múltiples arquitecturas.

1999: Lanzamiento de Linux 2.2.

2001: Lanzamiento de Linux 2.4, con mejoras significativas en el soporte de hardware y sistemas de archivos.

2003: Lanzamiento de Linux 2.6. Esta serie trajo muchas mejoras y características, como udev, SELinux, y soporte para el sistema de archivos ext4.

B) ¿Cuáles son sus funciones principales?

**Gestión de memoria**: asigna espacio en RAM a programas y datos, y gestiona el intercambio de memoria entre RAM y el disco cuando es necesario.

**Gestión de procesos**: controla la ejecución de procesos, garantizando el uso adecuado del CPU. **Gestión de dispositivos**: proporciona una interfaz para interactuar con hardware, como discos duros, pantallas, teclados, entre otros.

**Gestión de sistemas de archivos**: se encarga de las operaciones de lectura/escritura en los sistemas de archivos.

**Proporcionar una capa de abstracción**: permite que las aplicaciones se comuniquen con el hardware sin necesidad de conocer detalles específicos del mismo.

C) ¿Cuál es la versión actual? ¿Cómo se definía el esquema de versionado del Kernel en versiones anteriores a la 2?4? ¿Qué cambió en el versionado se impuso a partir de la versión 2.6?

A fecha de agosto de 2022, la última subversión disponible de este núcleo es Linux 5.19

Antes de 2.4: El esquema de versionado era más simple y menos estructurado.

Desde 2.6: Linus Torvalds introdujo un nuevo esquema de versionado. Las versiones estables del kernel tenían tres números (por ejemplo, 2.6.35). Si una versión tenía cuatro números, el último se refería a correcciones de seguridad y pequeñas actualizaciones. Posteriormente, Linus decidió simplificar el versionado principal pasando directamente a la versión 3.0 en lugar de 2.6.40.

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

Sí, es posible. Esto se hace comúnmente para pruebas o para tener un kernel de respaldo en caso de problemas con versiones más nuevas. Al arrancar, el gestor de arranque (por ejemplo, GRUB) permite seleccionar qué kernel se desea usar.

E) ¿Dónde se encuentra ubicado dentro del File System?

Generalmente, los archivos del kernel y sus imágenes se encuentran en el directorio /boot.

F) ¿El Kernel de GNU/Linux es monolítico? Justifique.

Sí, el kernel de GNU/Linux se clasifica como un kernel **monolítico**. Esto significa que todo el código del kernel, incluido el controlador de dispositivos, el planificador, la gestión de memoria, etc., se ejecuta en el mismo espacio de direcciones, es decir, en modo kernel. Sin embargo, tiene características de kernel micro (como módulos del kernel) que permiten cargar y descargar código en tiempo de ejecución. Estos módulos pueden ser controladores de dispositivos, sistemas de archivos y otras funciones del kernel, lo que permite una cierta modularidad y flexibilidad, pero el núcleo principal sigue siendo monolítico.

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

A) ¿Qué es?

El Shell (intérprete de comandos) es una interfaz de usuario para el sistema operativo que permite a los usuarios interactuar con el sistema a través de comandos escritos. Estos comandos se interpretan y se traducen a las llamadas de sistema que realiza el kernel.

B) ¿Cuáles son sus funciones?

Interactuar con el usuario: Recibir y procesar los comandos ingresados.

Manipulación de archivos y directorios.

Ejecutar programas y procesos.

Redirección de entradas y salidas.

Proporcionar programación de scripts, permitiendo automatizar tareas.

Gestionar procesos en segundo plano (background).

Proporcionar variables de entorno y de shell.

C) Mencione al menos 3 intérpretes de comandos que posee *GNU/Linux* y compárelos entre ellos.

**Bash (Bourne Again SHell**): Es el shell más popular en sistemas GNU/Linux. Es una extensión del Bourne Shell original y proporciona programación de scripts avanzada, completado de comandos, y es conocido por su versatilidad y poder.

**Tcsh (TENEX C Shell):** Basado en el C Shell, tcsh ofrece características similares al C Shell pero con mejoras, como completado de comandos y edición de línea de comandos.

**Zsh (Z Shell):** Ofrece muchas de las mismas características que bash, pero con algunas extensiones y mejoras, especialmente en la programación de scripts y en el completamiento de comandos. Es conocido por su flexibilidad y personalización.

**Comparación**: Bash es probablemente el más universalmente adoptado y es el shell predeterminado en muchas distribuciones de Linux. Tcsh y zsh, mientras que proporcionan características únicas, no son tan ampliamente utilizados como Bash. Zsh, sin embargo, ha ganado popularidad recientemente por su extensibilidad y características avanzadas.

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

Los comandos en un sistema GNU/Linux generalmente se ubican en varias ubicaciones del sistema de archivos, incluidos los siguientes directorios:

**/bin**: Comandos esenciales que el sistema necesita para arrancar y funcionar en modo de usuario único.

**/usr/bin**: Comandos no esenciales para usuarios básicos.

**/sbin**: Comandos esenciales para el sistema, generalmente destinados a ser utilizados por el administrador.

**/usr/sbin**: Comandos no esenciales para el sistema, también destinados para el administrador.

La variable de entorno **PATH** determina en qué directorios el shell buscará para ejecutar un comando. Se puede ver el valor actual de PATH con el comando $PATH.

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

El Shell y el Kernel son dos componentes distintos del sistema operativo. El Kernel es el núcleo del sistema operativo, gestionando los recursos del sistema y proporcionando servicios a los programas. El Shell, por otro lado, es simplemente una interfaz que permite a los usuarios interactuar con el kernel y el sistema en general. Mientras que el Kernel interactúa directamente con el hardware y gestiona recursos, el Shell se comunica con el Kernel mediante llamadas de sistema.

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 diferente para cada usuario. Esto se define en el archivo /etc/passwd. Cada línea de este archivo define un usuario y, al final de cada línea, hay una referencia al shell predeterminado para ese usuario.

Por ejemplo, una línea podría terminar en /bin/bash, indicando que Bash es el shell predeterminado para ese usuario.

Solo el administrador del sistema (root) o alguien con permisos adecuados puede cambiar el shell predeterminado de un usuario editando directamente este archivo o utilizando comandos como chsh. Un usuario sin privilegios de administrador no puede cambiar el shell de otro usuario, pero sí puede solicitar el cambio de su propio shell.

1. Sistema de Archivos *(File System)*:
   1. ¿Qué es?
   2. Mencione sistemas de archivos soportados por *GNU/Linux*.
   3. ¿Es posible visualizar particiones del tipo **FAT** y **NTFS** en *GNU/Linux*?
   4. ¿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**?

A) ¿Qué es?

Un **sistema de archivos** (File System) es una estructura lógica que permite a un sistema operativo almacenar, organizar, nombrar y recuperar datos en un dispositivo de almacenamiento, como un disco duro, SSD, USB, entre otros. Define cómo se almacenan los datos en estos dispositivos y cómo se organizan para que el sistema y las aplicaciones puedan acceder a ellos.

B) Mencione sistemas de archivos soportados por GNU/Linux.

GNU/Linux soporta una amplia variedad de sistemas de archivos, entre los que se incluyen:

**ext2, ext3, ext4**: Son sistemas de archivos nativos de Linux, siendo ext4 el más reciente y ampliamente utilizado.

**XFS**: Un sistema de archivos de alto rendimiento.

**Btrfs (B-tree file system)**: Un sistema de archivos moderno que ofrece características avanzadas como instantáneas y almacenamiento optimizado.

**ReiserFS**: Conocido por ser eficiente en la manipulación de pequeños archivos.

**JFS (Journaled File System)**: Desarrollado por IBM, ofrece registro para garantizar la integridad de los datos.

**FAT (FAT16, FAT32)**: Ampliamente utilizado en dispositivos de almacenamiento USB.

**NTFS**: Sistema de archivos de Windows.

**ZFS**: Desarrollado originalmente por Sun Microsystems para Solaris, ha sido portado a Linux aunque no está incluido en el kernel principal debido a problemas de licencia.

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

Sí, es posible. GNU/Linux puede montar y acceder a particiones FAT (incluidas FAT16 y FAT32) y NTFS. Existen herramientas y drivers, como ntfs-3g, que permiten leer y escribir en particiones NTFS desde Linux.

D) ¿Cuál es la estructura básica de los File System en GNU/Linux? Mencione los directorios más importantes e indique qué tipo de información se encuentra en ellos.

GNU/Linux sigue la jerarquía estándar del sistema de archivos (Filesystem Hierarchy Standard, FHS). Algunos directorios importantes son:

**/bin**: Binarios esenciales que se necesitan para arrancar el sistema.

**/boot**: Contiene los archivos de arranque, como el kernel y el inicializador del sistema.

**/dev**: Archivos de dispositivo que representan hardware y dispositivos periféricos.

**/etc**: Archivos de configuración del sistema.

**/home**: Directorios personales de los usuarios.

**/lib**: Bibliotecas compartidas y módulos del kernel.

**/mnt**: Punto de montaje temporal para sistemas de archivos.

**/opt**: Software adicional y paquetes de aplicaciones.

**/proc**: Sistema de archivos virtual que proporciona información sobre procesos y recursos del sistema.

**/root**: Directorio personal del usuario root.

**/sbin**: Binarios esenciales del sistema, principalmente para tareas de administración.

**/tmp**: Archivos temporales.

**/usr**: Aplicaciones y archivos compartidos.

**/var**: Datos variables como registros y bases de datos.

¿A qué hace referencia la sigla FHS?

**FHS** hace referencia a "Filesystem Hierarchy Standard". Es un estándar que define la estructura y disposición de directorios en sistemas operativos UNIX-like, como Linux. Su objetivo es mantener la consistencia y la interoperabilidad entre diferentes distribuciones y aplicaciones.

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

A) Definición. Tipos de particiones. Ventajas y Desventajas.

**Definición**: Una partición es una subdivisión del espacio de almacenamiento en un disco duro o SSD. Las particiones funcionan como contenedores independientes, que permiten instalar sistemas operativos, almacenar archivos o separar tipos de datos.

**Tipos de particiones**:

**Primaria**: Puede contener un sistema operativo. La mayoría de los sistemas de partición permiten hasta cuatro particiones primarias.

**Extendida**: Se usa como contenedor para particiones lógicas. Sólo puede existir una partición extendida por disco y no puede contener directamente archivos, sólo particiones lógicas.

**Lógica**: Reside dentro de una partición extendida y puede contener datos o un sistema operativo.

**Ventajas**:

Permite instalar múltiples sistemas operativos en un solo disco.

Facilita la organización y separación de datos.

Puede mejorar el rendimiento y la administración del espacio.

**Desventajas**:

Aumenta la complejidad en la administración del disco.

Existe el riesgo de perder datos si no se gestionan adecuadamente.

Hay un límite en la cantidad de particiones primarias.

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

Las particiones en GNU/Linux se identifican generalmente por su nombre de dispositivo:

**IDE**: Estos discos suelen ser representados como /dev/hd[a-d], donde la letra final indica el dispositivo (a es el primero, b es el segundo, etc.). Las particiones son numeradas, como /dev/hda1,/dev/hda2,etc.  
**SCSI y SATA**: Estos se representan como /dev/sd[a-z]. Al igual que con los discos IDE, las particiones se numeran, como /dev/sda1, /dev/sda2, etc.

C) ¿Cuántas particiones son necesarias como mínimo para instalar *GNU/Linux*? Nómbrelas indicando tipo de partición, identificación, tipo de File System y punto de montaje.

Como mínimo, se necesita una partición para la raíz (/). Sin embargo, en muchas instalaciones, se recomienda tener una partición separada para el swap.

**/ (raíz):** Primaria, Tipo de FS: ext4 (comúnmente), Punto de montaje: /.

**swap**: Lógica o primaria, Tipo de FS: swap. No tiene punto de montaje como tal, pero se usa como memoria virtual.

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

**Estación de trabajo general**: Una partición para /, una para /home (para separar datos personales del sistema) y una para swap.

**Servidor web**: Una partición para /, una para /var/www (para almacenar sitios web), una para /var/log (para registros) y una para swap.  
**Servidor de bases de datos**: Una partición para /, una para /var/lib/mysql o /var/lib/postgresql (dependiendo del sistema de DB), y una para swap.

E) ¿Qué tipo de software para particionar existe? Menciónelos y compare.

**fdisk**: Herramienta de línea de comandos para administrar particiones en discos. Es potente pero no es amigable para principiantes.

**gparted**: Interfaz gráfica que proporciona una forma fácil de gestionar particiones. Es uno de los más populares y amigables para usuarios sin experiencia.

**parted**: Herramienta de línea de comandos que puede manejar discos grandes y particiones con diferentes tipos de tablas.

**cfdisk**: Similar a fdisk, pero con una interfaz un poco más amigable.

Comparación:

fdisk y cfdisk son excelentes para usuarios que están familiarizados con la línea de comandos y necesitan una herramienta rápida y eficaz.

gparted es ideal para aquellos que prefieren una interfaz gráfica y requieren una visualización clara de las operaciones de particionamiento.

parted es poderoso y versátil, capaz de manejar una variedad de tipos de tablas de particiones y es excelente para scripts y automatización.

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

A) ¿Qué es el BIOS? ¿Qué tarea realiza?

**BIOS** (Basic Input/Output System) es el software incorporado en una pequeña memoria en la placa base de la computadora que se ejecuta al encenderla. Su función principal es inicializar y probar el hardware del sistema y cargar un sistema operativo desde un dispositivo de almacenamiento.

B) ¿Qué es UEFI? ¿Cuál es su función?

**UEFI** (Unified Extensible Firmware Interface) es un estándar moderno que reemplaza al BIOS. Proporciona una interfaz más avanzada y un conjunto de características, como el arranque seguro y el soporte para particiones de disco más grandes que 2 TB.

C) ¿Qué es el MBR? ¿Qué es el MBC?

**MBR** (Master Boot Record) es un sector especial en dispositivos de almacenamiento que contiene el cargador de arranque para el sistema operativo y la tabla de particiones.

El **MBC** es un pequeño código que permite arrancar el SO.

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

**GPT** (GUID Partition Table) es un estándar de diseño de tabla de particiones que sustituye al antiguo formato MBR. Es parte del estándar UEFI y permite manejar discos más grandes, así como más particiones. **GPT** usa identificadores únicos globales (GUIDs) para definir particiones y es más robusto en términos de integridad de datos.

E) ¿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 programa que carga el sistema operativo o da la opción de seleccionar entre varios sistemas operativos en un sistema multiarranque. Algunos gestores de arranque conocidos son:

**GRUB** (GRand Unified Bootloader): Es el gestor de arranque más común en distribuciones Linux.

**Windows Boot Manager**: Utilizado por Windows.

**LILO** (Linux Loader): Un cargador de arranque anteriormente popular para Linux, pero ha sido en gran parte reemplazado por GRUB.

Estos gestores de arranque generalmente se instalan en el MBR de un disco o en una partición EFI en sistemas UEFI.

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

La computadora se enciende y el BIOS/UEFI se inicializa.

El BIOS/UEFI realiza un POST (Power-On Self-Test) para comprobar el hardware.

Luego busca un dispositivo de arranque válido.

Se carga y se ejecuta el gestor de arranque desde el MBR o la partición EFI.

El gestor de arranque carga el núcleo del sistema operativo.

El núcleo del sistema operativo inicializa el hardware y carga los controladores.

Se inicia el sistema init (o su reemplazo, como systemd en muchas distribuciones Linux modernas) que continúa con el proceso de inicio, como montar sistemas de archivos e iniciar servicios.

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

BIOS/UEFI se inicializa y encuentra el gestor de arranque.

GRUB (u otro gestor de arranque) presenta opciones y carga el núcleo Linux seleccionado.

El núcleo inicializa el hardware y carga los controladores de dispositivo.

Se inicia el sistema init (por ejemplo, systemd o SysVinit).

Se montan los sistemas de archivos.

Se inician los servicios y demonios según la configuración.

Se presenta la pantalla de inicio de sesión para que el usuario se autentique.

H) ¿Cuáles son los pasos que se suceden en el proceso de parada (shutdown) de GNU/Linux?

Se envía una señal para terminar o detener todos los procesos y servicios en ejecución.

Se desmontan todos los sistemas de archivos.

Se detienen los servicios esenciales, como el gestor de redes.

Se apaga o reinicia el hardware, dependiendo de la solicitud.

I) ¿Es posible tener en una PC GNU/Linux y otro Sistema Operativo instalado? Justifique.

Sí, es posible y es conocido como configuración de "doble arranque" o "multiarranque". Esto permite a los usuarios seleccionar el sistema operativo que desean usar al encender la computadora. La justificación es que algunos usuarios pueden necesitar software específico o características de un sistema operativo, pero también quieren las ventajas o la familiaridad de otro. Se instalan en particiones separadas y el gestor de arranque (como GRUB) permite seleccionar cuál sistema operativo arrancar.

1. Archivos:
   1. ¿Cómo se identifican los archivos en *GNU/Linux*?
   2. Investigue el funcionamiento de los editores **vi** y **mcedit**, y los comandos **cat** y **more**.
   3. Cree un archivo llamado “prueba.exe” en su directorio personal usando el vi. El mismo debe contener su número de alumno y su nombre.
   4. Investigue el funcionamiento del comando file. Pruébelo con diferentes archivos. ¿Qué diferencia nota?

A) ¿Cómo se identifican los archivos en GNU/Linux?

En GNU/Linux, los archivos no se identifican por una extensión de archivo (como .exe o .doc) como en algunos otros sistemas operativos, aunque se pueden usar extensiones por convención. En cambio, cualquier archivo puede ser cualquier cosa: un programa, un script, un documento, etc. Los archivos se identifican principalmente por sus permisos y por el tipo de archivo que el sistema detecta a través de su contenido y metadatos. El comando file es una herramienta que se utiliza para determinar el tipo de archivo basado en esos datos.

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

**vi**: Es un editor de texto muy conocido en ambientes UNIX y Linux. Tiene dos modos principales: el modo normal (donde puedes navegar y realizar comandos) y el modo de inserción (donde puedes editar el archivo). Se entra en el modo de inserción presionando i y se regresa al modo normal con Esc. Para guardar y salir, en modo normal, se escribe :wq.

**mcedit**: Es el editor de la suite Midnight Commander. Es un editor más amigable y visual que vi con una interfaz de usuario que recuerda a los programas de DOS.

**cat**: Este comando se utiliza para visualizar el contenido de un archivo en la terminal. Por ejemplo, cat archivo.txt mostrará el contenido de "archivo.txt".

**more**: Es una utilidad que permite ver el contenido de un archivo página por página. Es útil para archivos largos donde cat mostraría todo el contenido de una vez.

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

Para hacer esto en un sistema real, seguirías los siguientes pasos:

Abre la terminal.

Escribe vi prueba.exe y presiona Enter.

Presiona i para entrar en el modo de inserción.

Escribe tu número de alumno y tu nombre.

Presiona Esc para regresar al modo normal.

Escribe: wq y presiona Enter para guardar y salir.

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

El comando file examina el contenido de un archivo y, basado en ciertas heurísticas y la base de datos de "magic numbers", intenta determinar qué tipo de archivo es. Por ejemplo:

Copy code

file imagen.jpg

Podría devolver: "imagen.jpg: JPEG image data, JFIF standard 1.01"

Si pruebas file en diferentes archivos, notarás que el comando intentará identificar el tipo de archivo y puede proporcionar información adicional sobre él, como la codificación, si es un archivo de texto, o la resolución y tipo de archivo, si es una imagen.

1. Indique qué comando es necesario utilizar para realizar cada una de las siguientes acciones. Investigue su funcionamiento y parámetros más importantes:
   1. Cree la carpeta ISO2017
   2. Acceda a la carpeta (cd)
   3. Cree dos archivos con los nombres iso2017-1 e iso2017-2 (touch)
   4. Liste el contenido del directorio actual (ls)
   5. Visualizar la ruta donde estoy situado (pwd)
   6. Busque todos los archivos en los que su nombre contiene la cadena “iso\*” (find)
   7. Informar la cantidad de espacio libre en disco (df)
   8. Verifique los usuarios conectado al sistema (who) (i) Acceder a el archivo iso2017-1 e ingresar Nombre y Apellido (j) Mostrar en pantalla las últimas líneas de un archivo (tail).

A) Cree la carpeta ISO2017

bash

mkdir ISO2017

mkdir: Comando para "make directory", es decir, crear directorios.

ISO2017: Es el nombre del directorio que quieres crear.

B) Acceda a la carpeta (cd)

bash

cd ISO2017

cd: Comando "change directory", permite cambiar el directorio actual al especificado.

ISO2017: Nombre del directorio al que quieres acceder.

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

bash

touch iso2017-1 iso2017-2

touch: Comando utilizado principalmente para crear archivos vacíos, pero también puede actualizar las fechas de acceso y modificación de archivos existentes.

iso2017-1 y iso2017-2: Nombres de los archivos que deseas crear.

D) Liste el contenido del directorio actual (ls)

bash

ls

ls: Comando "list", muestra el contenido del directorio actual.

Se puede usar con varios modificadores, como -l para mostrar en formato largo, -a para mostrar archivos ocultos, etc.

E) Visualizar la ruta donde estoy situado (pwd)

bash

pwd

pwd: "print working directory", muestra el camino absoluto del directorio actual.

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

bash

find . -name "iso\*"

find: Herramienta para buscar archivos y directorios.

Representa el directorio actual. La búsqueda comenzará aquí y se extenderá a todos los subdirectorios.

-name: Especifica que se buscará por nombre de archivo.

"iso\*": Patrón de búsqueda; busca archivos que comienzan con "iso".

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

bash

df -h

df: Comando "disk free", muestra el uso del disco de sistemas de archivos.

-h: Modificador que hace que el tamaño se muestre en un formato "human readable" (legible por humanos, como 1K, 234M, 2G).

H) Verifique los usuarios conectados al sistema (who)

bash

who

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

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

bash

nano iso2017-1

nano: Es un editor de texto sencillo que opera en la terminal. Al ejecutar este comando, se abrirá el archivo en el editor, permitiéndote escribir o modificar su contenido. Una vez que hayas escrito tu Nombre y Apellido, puedes presionar Ctrl + O para guardar y Ctrl + X para salir.

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

bash

tail iso2017-1

tail: Este comando muestra las últimas líneas de un archivo.

Por defecto, tail muestra las últimas 10 líneas. Si deseas un número específico de líneas, puedes usar tail -n [número]. Por ejemplo, tail -n 5 iso2017-1 mostrará las últimas 5 líneas.

1. Investigue su funcionamiento y parámetros más importantes:
2. shutdown
3. reboot
4. halt
5. locate
6. uname
7. gmesg
8. lspci
9. at
10. netstat
11. mount
12. umount
13. head
14. losetup
15. write
16. mkfs
17. fdisk (con cuidado)

A)

shutdown: Apaga o reinicia el sistema.

-h: Apagar el sistema después de que se detenga.

-r: Reiniciar el sistema después de que se detenga.

now: Indica que el apagado/reinicio es inmediato.

B)

reboot: Reinicia el sistema. Esencialmente es un alias para shutdown -r now.

C)

halt: Detiene el sistema. En versiones modernas, es similar a shutdown -h now.

D)

locate: Busca archivos por nombre. Utiliza una base de datos que es actualizada con el comando updatedb.

Ejemplo: locate archivo.txt

E)

uname: Muestra información del sistema.

-a: Muestra toda la información.

r: Muestra la versión del kernel.

F)

dmesg: Muestra mensajes del kernel.

-w: Observa los mensajes del kernel en tiempo real.

-C: Limpia el búfer del registro del kernel.

G)

lspci: Lista todos los dispositivos PCI.

-v: Muestra información detallada.

H)

at: Programa comandos para ser ejecutados en una fecha y hora específica.

Ejemplo: echo "ls -l" | at 10:30 ejecutará ls -l a las 10:30.

I)

netstat: Muestra estadísticas de la red.

-a: Muestra todos los puertos y conexiones.

-t: Muestra conexiones TCP.

J)

mount: Monta sistemas de archivos.

Ejemplo: mount /dev/sda1 /mnt

K)

umount: Desmonta sistemas de archivos.

Ejemplo: umount /mnt

L)

head: Muestra las primeras líneas de un archivo.

-n: Número de líneas a mostrar.

M)

losetup: Configura y muestra dispositivos de bucle (loop devices).

-f: Muestra el primer dispositivo libre.

-d: Desasocia el dispositivo.

N)

write: Envía un mensaje a otro usuario.

Ejemplo: write usuario luego escribes el mensaje y presionas Ctrl+D para enviar.

O)

mkfs: Crea un sistema de archivos en un dispositivo.

Ejemplo: mkfs -t ext4 /dev/sda1 crea un sistema de archivos ext4 en el dispositivo /dev/sda1.

P)

fdisk: Herramienta para manipular particiones de disco. ¡Con cuidado!

-l: Lista las particiones

1. Investigue su funcionamiento y parámetros más importantes:
   1. Indique en qué directorios se almacenan los comandos mencionados en el ejercicio anterior.

* **shutdown**: /sbin/shutdown
* **reboot**: /sbin/reboot
* **halt**: /sbin/halt
* **locate**: /usr/bin/locate
* **uname**: /bin/uname
* **dmesg**: /bin/dmesg
* **lspci**: /usr/bin/lspci
* **at**: /usr/bin/at
* **netstat**: /usr/bin/netstat (o /bin/netstat en algunas configuraciones)
* **mount**: /bin/mount
* **umount**: /bin/umount
* **head**: /usr/bin/head
* **losetup**: /sbin/losetup
* **write**: /usr/bin/write
* **mkfs**: /sbin/mkfs
* **fdisk**: /sbin/fdisk

1. <https://www.debian.org/intro/about> [↑](#footnote-ref-1)