

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*:
   1. Mencione y explique las características más relevantes de *GNU/Linux*.
   2. Mencione otros sistemas operativos y compárelos con *GNU/Linux* en cuanto a los puntos mencionados en el inciso *a*.
   3. ¿Qué es **GNU**?
   4. Indique una breve historia sobre la evolución del proyecto *GNU*
   5. Explique qué es la multitarea, e indique si *GNU/Linux* hace uso de ella.
   6. ¿Qué es **POSIX**?

A- Es multiusuario

* Permite que varios usuarios accedan simultáneamente al sistema (Ej servidores)

 Es multitarea y multiprocesador

* Puede ejecutar varias tareas al mismo tiempo y permite Multicore

 Es altamente portable

* Puede ejecutarse en una gran variedad de hardware

 Posee diversos interpretes de comandos, de los cuales algunos son programables

* Linux ofrece varios intérpretes de comandos (Shell), como Bash, Zsh.
* Bash, es programables, lo que significa que los usuarios pueden crear scripts

 Permite el manejo de usuarios y permisos

* Permite que un superusuario controle los permisos de acceso a los archivos (Y los usuarios, con limites también pueden seleccionar quien puede editar un archivo)

 Todo es un archivo (hasta los dispositivos y directorios)

* En Linux, todo se maneja como si fuera un archivo (Los directorios son una lista de archivos)

 Cada directorio puede estar en una partición diferente (/temp, /home, etc.)

* Esto significa que puede tener directorios en particiones separadas mejorando la administración de espacio en disco y la seguridad del sistema.

 Es case sensitive

* Es sensible a mayúsculas y minúsculas

 Es código abierto

* Cualquiera puede ver como está hecho

B) Otros sistemas operativos: Windows

Es multiusuario SI

 Es multitarea y multiprocesador SI

 Es altamente portable NO, Windows está diseñado para funcionar en arquitecturas x86/x64

 Posee diversos interpretes de comandos SI (Comamand Prompt y PowerShell)

 Permite el manejo de usuarios y permisos SI (Es mas engorroso)

 Todo es un archivo NO (todo es un objeto)

Cada directorio puede estar en una partición diferente (/temp, /home, etc.) SI

 Es case sensitive NO

 Es codigo abierto NO

Mac-os

Es multiusuario NO

 Es multitarea y multiprocesador SI

 Es altamente portable NO, Windows está diseñado para funcionar en arquitecturas x86/x64

 Posee diversos interpretes de comandos NO

 Permite el manejo de usuarios y permisos NO

 Todo es un archivo NO

Cada directorio puede estar en una partición diferente (/temp, /home, etc NO

 Es case sensitive NO

 Es codigo abierto NO

C) GNU (GNU No es Unix) es un Sistema Operativo tipo Unix, pero libre y de codigo abierto

+ DATA: Unix es un sistema operativo y una familia de sistemas operativos de tipo multitarea y multiusuario que se desarrolló originalmente en los años 1960 en los laboratorios de investigación de AT&T (American Telephone and Telegraph) en los Estados Unidos. Fue concebido por Ken Thompson, Dennis Ritchie y otros colaboradores en los laboratorios Bell como una respuesta a la necesidad de un sistema operativo más eficiente y versátil para la programación y el uso en sistemas de cómputo

D) Historia:

 Iniciado por Richard Stallman en 1983 con el fin de crear un Unix libre (el sistema GNU)

 Para asegurar que el mismo fuera libre, se necesitó crear un marco regulatorio conocido como GPL (General Public License de GNU)

 En 1985, Stallman crea la FSF (Free Software Foundation), con el fin de financiar el proyecto GNU

 En 1990, GNU ya contaba con un editor de textos (Emacs), un compilador (GCC) y gran cantidad de bibliotecas que componen un Unix típico.

 Faltaba el componente principal → El Núcleo (Kernel)

Si bien ya se venía trabajando en un núcleo conocido como TRIX, es en 1988 que se decide abandonarlo debido a su complejidad (corre en hardware muy costoso)

 Se decide adoptar como base el núcleo MACH para crear GNU Hurd, el cual tampoco prospero

 Linus Torvalds ya venía trabajando desde 1991 en un Kernel denominado Linux, el cual se distribuiría bajo licencia GPL

 En el año 1992, Torvalds y Stallman deciden fusionar ambos proyectos, y nace GNU/Linux

 GNU/Linux pertenece al desarrollo del software libre

E) La multitarea es una capacidad de los sistemas operativos que permite ejecutar múltiples tareas o programas de forma aparentemente simultánea, incluso en sistemas con un solo procesador o núcleo de CPU. Esto significa que un sistema operativo con multitarea puede alternar rápidamente entre diferentes programas, asignándoles recursos de procesador y permitiendo que realicen progresos en paralelo.

Linux la posee.

F) La Portable Operating System Interface for Unix (POSIX) es un conjunto de estándares que define una interfaz común y compatible entre sistemas operativos tipo

El objetivo principal de POSIX es asegurar la portabilidad del software entre diferentes sistemas Unix

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 e indique cuáles son los objetivos del proyecto y una breve cronología del mismo
2. Una distribución es una customización de GNU/Linux formada por una versión de kernel y determinados programas con sus configuraciones
3. **Ubuntu**:
   * **Objetivo**: Ubuntu es conocida por su enfoque en la facilidad de uso y la accesibilidad. Está diseñada para ser una distribución amigable para los principiantes de Linux.
   * **Entorno de escritorio predeterminado**: Ubuntu utiliza el entorno de escritorio GNOME como su entorno de escritorio predeterminado
   * **Ciclo de lanzamiento**: Tiene versiones de soporte que reciben actualizaciones y soporte durante 5 años, así como versiones regulares con soporte a corto plazo.
4. **Fedora**:
   * **Objetivo**: Fedora se enfoca en ofrecer software de código abierto y las últimas tecnologías de manera estable y segura.

Es una distribución más orientada a usuarios avanzados y desarrolladores.

* + **Entorno de escritorio predeterminado**: Utiliza el entorno de escritorio GNOME de forma predeterminada, pero también ofrece opciones de otros entornos.
  + **Ciclo de lanzamiento**: Se actualiza aproximadamente cada 6 meses. Las versiones son conocidas por incluir tecnologías y características de vanguardia.

1. **Debian**:
   * **Objetivo**: Debian es una de las distribuciones más antiguas y se enfoca en la estabilidad y el software 100% de código abierto. Es conocida por su proceso de desarrollo deliberado y su compromiso con el software libre.
   * **Entorno de escritorio predeterminado**: Debian ofrece varias opciones de entornos de escritorio, pero no tiene uno predeterminado. Los usuarios pueden elegir el que deseen.
   * **Ciclo de lanzamiento**: Tiene un ciclo de lanzamiento más conservador y estable. Las versiones son conocidas por tener un enfoque en la estabilidad a largo plazo.
2. **Arch Linux**:
   * **Objetivo**: Arch Linux es una distribución extremadamente personalizable, orientada a usuarios avanzados. Se centra en la simplicidad y el control del usuario sobre la configuración.
   * **Entorno de escritorio predeterminado**: No tiene un entorno de escritorio predeterminado. Los usuarios deben configurar su sistema según sus preferencias.
   * **Modelo de actualización continua**: Arch Linux sigue un modelo de actualización continua.

Diagrama

Descripción generada automáticamente

C) Debian es una distribución de GNU/Linux que se caracteriza por su compromiso con los principios del software libre y su enfoque en la estabilidad y la calidad del sistema. Es una de las distribuciones más antiguas y respetadas en el mundo del software libre. Los objetivos principales del proyecto Debian son:

1. **Software Libre**: El objetivo central de Debian es promover y defender el software libre. Se compromete a utilizar y distribuir software que cumpla con las definiciones de software libre según las pautas de la Free Software Foundation (FSF).
2. **Estabilidad**: Debian se enfoca en proporcionar un sistema estable y confiable. Esto se logra a través de un proceso de prueba y desarrollo riguroso antes de lanzar nuevas versiones.
3. **Universalidad**: Debian se esfuerza por ser una distribución universal que funcione en una amplia variedad de arquitecturas de hardware. Esto incluye no solo sistemas x86/x64, sino también sistemas ARM, MIPS, PowerPC y otros.
4. **Facilidad de Uso**: Aunque Debian tiene una reputación de ser más orientada a usuarios avanzados, también se preocupa por la facilidad de uso. Ofrece varios entornos de escritorio y herramientas que hacen que la experiencia de usuario sea más accesible.
5. **Mantener el Control del Usuario**: Debian otorga a los usuarios un alto nivel de control y personalización sobre su sistema. Permite a los usuarios elegir componentes específicos durante la instalación y brinda flexibilidad en la configuración.
6. **Comunidad y Desarrollo Colaborativo**: Debian es un proyecto comunitario en el que muchas personas de todo el mundo contribuyen al desarrollo, prueba y mantenimiento de la distribución. El proyecto Debian es conocido por su énfasis en la colaboración y la toma de decisiones democráticas.

Cronología breve del proyecto Debian:

* **1993**: Debian se inicia oficialmente el 16 de agosto por Ian Murdock, quien publica la primera versión del "Manifiesto Debian" y establece los principios fundacionales del proyecto.
* **1994**: Se lanza la primera versión de Debian, Debian 0.01.
* **1996**: Debian 1.0 (Buzz) se lanza y se establece el sistema de gestión de paquetes Debian APT (Advanced Package Tool).
* **1998**: Debian 2.0 (Hamm) se lanza con un sistema de paquetes más avanzado y mejoras en la facilidad de instalación.
* **2000**: Debian 2.2 (Potato) introduce el Proyecto de Directrices de Software Libre de Debian.
* **2002**: Debian 3.0 (Woody) se lanza con una arquitectura de múltiples ramas, incluyendo la rama estable y la rama de pruebas.
* **2005**: Debian 3.1 (Sarge) se lanza, marcando la primera versión oficial con soporte para sistemas AMD64.
* **2007**: Debian 4.0 (Etch) se lanza con mejoras en la gestión de paquetes y mejor soporte para sistemas portátiles.
* **2013**: Debian 7.0 (Wheezy) se lanza con mejoras en el rendimiento y una mayor integración con Systemd.
* **2017**: Debian 9.0 (Stretch) se lanza con mejoras en la seguridad y la compatibilidad con arquitecturas de 64 bits.
* **2019**: Debian 10.0 (Buster) se lanza con mejoras en la instalación y una mayor compatibilidad con hardware moderno.
* **2021**: Debian 11.0 (Bullseye) se lanza, ofreciendo actualizaciones en la gestión de paquetes y nuevas versiones de software.

El proyecto Debian ha mantenido su compromiso con el software libre y ha influido significativamente en el mundo de Linux y el software libre a lo largo de su historia.

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*.
2. Los tres componentes fundamentales de un sistema GNU/Linux son: El núcleo (Kernel), El interprete de comandos y el Sistema de Archivos (Opcional el Shell)
3. Paquete de software que permite diferenciar una distribucion de otra.

 Editores de texto: vi ,Emacs, joe (Editor de TXT)

 Herramientas de networking: wireshark, tcpdump (Red)

 Paquetes de oficina:  OpenOffice (Editor de EXCEL)

 Iterface gr´aficas: GNOME / CINNAMON , KDE , LXDE (Escritorio)

1. *Kernel*:
   1. ¿Qué es? Indique una breve reseña histórica acerca de la evolución del Kernel de

*GNU/Linux*.

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

1. También conocido como núcleo, es el encargado de ejecutar programas y gestionar dispositivos de hardware

 En 1991 Linus Torvalds inicia la programacion de un Kernel Linux basado en Minix (clon de Unix desarrollado por Tenembaum en 1987 con el fin de crear un S.O. de uso didáctico)

 El 5 de octubre de 1991, se anuncia la primera versión “oficial” de Linux (0.02)

 En 1992 se combina su desarrollo con GNU, formando GNU/Linux

 La versión 1.0 apareció el 14 de marzo de 1994

1. Sus funciones mas importantes son la administración de memoria, CPU y la E/S
2. La ultima versión estable es la 6.4.11 (agosto de 2023)

Antes de la versión 2.4, el versionado del kernel de Linux seguía un esquema de tres números:

* El primer número representaba la versión principal del kernel.
* El segundo número representaba la versión secundaria o versión menor, que indicaba mejoras y adiciones significativas.
* El tercer número representaba la revisión o versión de parche, que se utilizaba para lanzar correcciones de errores y actualizaciones menores.

Nomenclatura a partir de 2.4:

A: Denota versión. Cambia con menor frecuencia. En 1994 (versión 1.0) y en 1996 (versión 2.0)

B: Denota mayor revisión.

C: Denota menor revisión. Solo cambia cuando hay nuevos drivers o características

D: Cambia cuando se corrige un grave error sin agregar nueva funcionalidad

Antes de la versión 2.6, los números impares indicaban desarrollo, los pares producción

1. SI, Cuando inicies la máquina, normalmente se te dará la opción de seleccionar qué kernel deseas utilizar en el gestor de arranque. Esto te permite elegir entre las versiones instaladas del kernel según tus necesidades o preferencias.
2. El código fuente del kernel instalado en nuestro sistema se encuentra normalmente en /usr/src/linux .
3. Es un núcleo monolítico hibrido:

 Un kernel monolítico es un tipo de diseño de kernel en el que la mayoría de las funcionalidades del sistema operativo se ejecutan en un solo espacio de memoria y como una sola

 Los drivers y código del Kernel se ejecutan en modo privilegiado

 Lo que lo hace hibrido es la capacidad de cargar y descargar funcionalidad a través de módulos

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?
2. El **Shell** es una herramienta del sistema operativo que sirve de mediador entre el usuario y el núcleo del sistema operativo
3. Funciones del Shell: Ejecuta programas a partir del ingreso de comandos, Control de Procesos, Automatización de Tareas, Control de Usuarios y Grupos, Personalización
4. Bourne Shell (sh): básico, Shell de inicio

Korn Shell (ksh), Avanzado, mejor control y capacidad de scripting

Bourne Again Shell (BASH), Superior, Además de lo anterior, soporta historial de comandoS

1. Comandos propios del shell: Los comandos propios del shell, como **cd**, **ls**, **pwd**, **echo** y otros, generalmente están ubicados en directorios específicos del sistema. Estos comandos suelen ser archivos ejecutables ubicados en directorios estándar como **/bin**, **/usr/bin** o **/sbin**, dependiendo del sistema y la distribución. Comandos externos al shell: Los comandos externos al shell pueden ubicarse en varios lugares, y el sistema utiliza una variable de entorno llamada **PATH** para buscar estos comandos en diferentes directorios. La variable **PATH** contiene una lista de directorios separados por dos puntos (**:**) donde el sistema buscará comandos cuando se ejecute un programa desde la línea de comandos. Puedes ver el valor de la variable **PATH** escribiendo **echo $PATH** en la línea de comandos.

Por lo general, los comandos de terceros y programas instalados por el usuario se encuentran en directorios como **/usr/local/bin**, **/opt**, **/usr/sbin**, **/usr/local/sbin**, y otros, dependiendo de cómo se configure el sistema y dónde se instalen los programas.

E)El Shell no es parte del Kernel de GNU/Linux porque cumple funciones muy diferentes en el sistema operativo y están diseñados para tareas distintas.

F)Si, Esto se hace configurando la entrada del intérprete de comandos en el archivo de configuración del usuario llamado ".bashrc" (para Bash, el Shell predeterminado en la mayoría de los sistemas), cualquier usuario que tenga permisos puede cambiar su Shell, el su puede cambiárselos a todos.

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 directo- rios más importantes e indique qué tipo de información se encuentra en ellos. ¿A qué hace referencia la sigla **FHS**?
2. Es la manera en que Linux organiza la el almacenamiento de los archivos
3. El adoptado por GNU/Linux es el Extended (v2, v3, v4)
4. Si: Para ver particiones FAT sudo mount -t vfat /dev/sdb1 /mnt/fat

Para ver particiones NTFS sudo apt-get install ntfs-3g

sudo mount -t ntfs-3g /dev/sdb1 /mnt/ntfs

1. Directorios mas importantes según la estructura básica que sigue el standard FHS (Filesystem Hierarchy Standard)

 / Tope de la estructura de directorios. Es como el C:\

 /home Se almacenan archivos de usuarios (Mis documentos)

 /var Información que varía de tamaño (logs, BD, spools)

 /etc Archivos de configuración

 /bin Archivos binarios y ejecutables

 /dev Enlace a dispositivos

 /usr Aplicaciones de usuario

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**).

* 1. ¿Cuántas particiones son necesarias como mínimo para instalar *GNU/Linux*? Nómbre- las indicando tipo de partición, identificación, tipo de File System y punto de montaje.
  2. Ejemplifique diversos casos de particionamiento dependiendo del tipo de tarea que se deba realizar en su sistema operativo.
  3. ¿Qué tipo de software para particionar existe? Menciónelos y compare.

1. Es una forma de dividir lógicamente el disco físico.

Debido al tamaño acotado en el MBR para la tabla de particiones:

 Se restringe a 4 la cantidad de particiones primarias

 3 primarias y una extendida con sus respectivas particiones lógicas

Las ventajas son que se pueden separar los datos para tener múltiples sistemas operativos, aislar los problemas (en caso de que ocurran en la partición A, no afecta ala B), y tener mayor seguridad y org.

Las desventajas son que pueden ocurrir Fragmentaciones, dificultad de gestión y desperdicio de espacio

1. B) **Tipo de dispositivo ([tipo\_de\_dispositivo]):** Este componente indica el tipo de dispositivo de almacenamiento. Algunos ejemplos comunes incluyen:
   * **hda** o **sda**: Disco duro IDE o SATA (el primer disco se llama **a**, el segundo **b**, y así sucesivamente).
   * **sdb**, **sdc**, etc.: Discos duros adicionales SATA o SCSI.
   * **hdb**, **hdc**, etc.: Discos duros IDE adicionales.
2. **Letra del dispositivo ([letra\_del\_dispositivo]):** Esta letra representa la interfaz del dispositivo. En el caso de los discos duros IDE, la letra es una letra del alfabeto (por ejemplo, **a**, **b**, **c**, etc.). En el caso de los discos duros SATA y SCSI, la letra no se utiliza en la notación.
3. **Número de partición ([número\_de\_partición]):** Este componente indica el número de la partición en el dispositivo. Las particiones se numeran secuencialmente a partir de 1. Por ejemplo, **/dev/sda1** representa la primera partición en el primer disco SATA.

Entonces, aquí hay algunos ejemplos de cómo se identifican las particiones en GNU/Linux en función del tipo de dispositivo:

* Primer disco duro IDE con tres particiones: **/dev/hda1**, **/dev/hda2**, **/dev/hda3**.
* Segundo disco duro SATA con dos particiones: **/dev/sdb1**, **/dev/sdb2**.
* Tercer disco duro SCSI con una partición: **/dev/sdc1**.

C) Como mínimo es necesario una partición (para el /), Es recomendable crear al menos 2 (/ y SWAP)

D) **1. Uso general en una computadora de escritorio o portátil:**

* **Partición raíz ("/"):** Esta es la partición principal y debe ser lo suficientemente grande para contener el sistema operativo y las aplicaciones. Por ejemplo, 30-50 GB podría ser adecuado.
* **Partición de intercambio (swap):** Se recomienda tener una partición de intercambio igual o ligeramente superior al tamaño de la RAM para el uso en caso de hibernación o falta de memoria.
* **Partición de inicio ("/boot"):** En sistemas con múltiples núcleos o si se planea cifrar el sistema, puede ser útil tener una pequeña partición de inicio (alrededor de 500 MB).

**2. Servidor web:**

* **Partición raíz ("/"):** Al igual que en el caso de uso general, una partición raíz para el sistema y las aplicaciones es necesaria.
* **Partición de datos ("/var/www"):** Para almacenar los archivos del sitio web y los datos de la aplicación.
* **Partición de registro ("/var/log"):** Para almacenar registros de actividad del servidor.

**3. Servidor de bases de datos:**

* **Partición raíz ("/"):** Como siempre, se necesita una partición raíz.
* **Partición de datos de la base de datos ("/var/lib/mysql"):** Para almacenar los archivos de datos de la base de datos.
* **Partición de registros de la base de datos ("/var/log/mysql"):** Para almacenar registros de la base de datos.

1. Para crear las particiones, se utiliza software denominado particionador.

Existen 2 tipos:  Destructivos: permiten crear y eliminar particiones (fdisk)

 No destructivo: permiten crear, eliminar y modificar particiones (fips, gparted) ← (generalmente las distribuciones permiten hacerlo desde la interfaz de instalación)

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/Li- nux*?
   9. ¿Es posible tener en una PC *GNU/Linux* y otro Sistema Operativo instalado? Justifi- que.
2. En las arquitecturas x86, el BIOS (Basic I/O System) es el responsable de iniciar la carga del SO a trav´es del MBC
3. UEFI (Unified Extensible Firmware Interface) es un estándar de firmware que reemplaza gradualmente al tradicional BIOS (Basic Input/Output System) en las computadoras modernas. Su función principal es proporcionar un entorno de firmware más avanzado y versátil que el BIOS, con el objetivo de mejorar el proceso de arranque y la interoperabilidad de hardware y software en las computadoras
4. **Master Boot Record (MBR):** En informática, MBR es la abreviatura de "Master Boot Record". Es una pequeña área de almacenamiento en el primer sector de un disco duro (cilindro 0, cabeza 0, sector 1). El MBR contiene información crítica para el proceso de inicio del sistema operativo en una computadora. Incluye un programa de inicio y una tabla de particiones que describe la estructura de partición del disco.

El término **"Master Boot Code**" (Código Maestro de Arranque, en español) se refiere a la parte del Master Boot Record (MBR) que contiene instrucciones de código de máquina específicas para iniciar el proceso de arranque de una computadora. El MBR es la primera sección de un dispositivo de almacenamiento, como un disco duro, y se encuentra en el sector de arranque (sector 0) de ese dispositivo. El Código Maestro de Arranque (Master Boot Code) es un programa de inicio muy pequeño que se ejecuta cuando se inicia la computadora y desempeña un papel esencial en el proceso de arranque del sistema operativo.

El tamaño del MBR coincide con el tamaño estándar de sector, 512 bytes:

 Los primeros bytes corresponden al Master Boot Code (MBC)

 A partir del byte 446 está la tabla de particiones. Es de 64 bytes

 Al final existen 2 bytes libres o para firmar el MBR

1. Utiliza el sistema GPT (GUID partition table) para solucionar limitaciones del MBR, como la cantidad de particiones , GPT especifica la ubicación y formato de la tabla de particiones en un disco duro. Puede verse como una sustitución del MBR
2. Un "Gestor de Arranque" es un programa o software que se utiliza para seleccionar y cargar un sistema operativo específico cuando se inicia una computadora. La funcionalidad principal de un gestor de arranque es permitir al usuario o al sistema determinar qué sistema operativo debe arrancar

**GRUB (GRand Unified Bootloader):** GRUB es un gestor de arranque muy popular en sistemas basados en Linux.

**Windows Boot Manager:** En sistemas Windows, el Administrador de Arranque de Windows (Windows Boot Manager) se utiliza para gestionar el inicio de múltiples versiones de Windows en una computadora

Los gestores de arranque suelen instalarse en la partición de arranque principal de la unidad de almacenamiento del sistema, como el MBR (Master Boot Record) en sistemas BIOS o la partición EFI en sistemas UEFI

1.  En las arquitecturas x86, el BIOS (Basic I/O System) es el responsable de iniciar la carga del SO a través del MBC:

 Esta grabado en un chip (ROM, NVRAM)

 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

 Luego pasa el control al proceso init

 El proceso de arranque se ve como una serie de pequeños programas encadenaos

1. 1 Se empieza a ejecutar el código del BIOS

2 El BIOS ejecuta el POST

3 El BIOS lee el sector de arranque (MBR)

4 Se carga el gestor de arranque (MBC)

5 El bootloader carga el kernel y el initrd

6 Se monta el initrd como sistema de archivos raíz y se inicializan componentes esenciales (ej.: scheduler)

7 El Kernel ejecuta el proceso init y se desmonta el initrd

8 Se lee el /etc/inittab

9 Se ejecutan los scripts apuntados por el runlevel 1

10 El final del runlevel 1 le indica que vaya al runlevel por defecto

11 Se ejecutan los scripts apuntados por el runlevel por defecto 12 El sistema está listo para usarse

1. Cierre de sesiones de usuario: El sistema primero notifica a los usuarios que están conectados al sistema que se va a realizar un apagado.

Cierre de aplicaciones: El sistema comienza a cerrar todas las aplicaciones y procesos en ejecución. Esto implica enviar señales de terminación (SIGTERM) a los procesos

Desmontaje de sistemas de archivos: El sistema de archivos se desmonta de manera ordenada. Esto implica asegurarse de que todos los sistemas de archivos estén en un estado coherente y que no haya escrituras pendientes en disco. El comando "sync" se utiliza para asegurar que los datos se escriban en el disco antes de continuar.

Apagado del sistema: Una vez que todas las aplicaciones se han cerrado y los sistemas de archivos se han desmontado, el sistema procede a apagar o reiniciar. Esto se puede hacer mediante comandos como "shutdown" o "reboot", y el sistema operativo finalmente envía señales al hardware para que se apague o reinicie.

Apagado del hardware: El hardware de la computadora se apaga o reinicia según la instrucción del sistema.

1. Si, ya lo hicimos
2. 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?
5. Por su nombre, por su ruta absoluta (Ej "/home/usuario/archivo.txt") o relativa "./archivo.txt"

B)

**Editor "vi":**

"vi" es un editor de texto en modo texto muy poderoso pero también notoriamente difícil de usar para principiantes debido a su interfaz de usuario basada en comandos. Aquí están algunos conceptos clave para entender cómo funciona "vi":

* **Modos de Edición:** "vi" tiene varios modos, siendo los dos más importantes el "modo normal" y el "modo de inserción". En el modo normal, puedes navegar por el archivo, buscar texto y ejecutar comandos. En el modo de inserción, puedes escribir y editar el texto. Para cambiar entre estos modos, puedes presionar "i" para entrar en el modo de inserción, y "Esc" para volver al modo normal.
* **Comandos:** En el modo normal, los comandos de "vi" se inician con una letra o caracteres específicos, como ":w" para guardar el archivo, ":q" para salir del editor y ":q!" para salir sin guardar cambios. Hay muchos otros comandos para buscar, copiar, pegar, eliminar, etc.
* **Guardado y Salida:** Para guardar un archivo, debes estar en el modo normal y usar ":w". Para salir del editor, debes estar en el modo normal y usar ":q". Para guardar y salir en una sola operación, puedes usar ":wq".

**Editor "mcedit" (Midnight Commander Editor):**

"mcedit" es parte del paquete Midnight Commander, que es un administrador de archivos de texto completo para la línea de comandos. "mcedit" es más fácil de usar en comparación con "vi" y tiene una interfaz de usuario basada en ventanas, similar a un editor de texto gráfico.

* **Interfaz Gráfica:** "mcedit" presenta una interfaz gráfica de usuario en la línea de comandos, lo que significa que puedes utilizar el mouse y las teclas de función para navegar y editar archivos. Esto lo hace más accesible para usuarios que no están familiarizados con los comandos de "vi".
* **Atajos de Teclado:** Aunque "mcedit" es más fácil de usar con el mouse, también es posible usar atajos de teclado. Puedes navegar por el menú superior para realizar acciones como abrir, guardar o buscar texto.
* **Funciones de Edición:** "mcedit" ofrece características de edición básicas, como copiar, pegar, buscar y reemplazar, todo ello utilizando combinaciones de teclas estándar. Esto hace que sea más intuitivo para los usuarios que están acostumbrados a los editores de texto gráficos.

**Comando "cat":**

* **Función Principal:** El comando "cat" (abreviatura de "concatenate") se utiliza principalmente para concatenar y mostrar el contenido de uno o varios archivos de texto en la salida estándar

**Comando "more":**

* **Función Principal:** El comando "more" se utiliza para mostrar el contenido de un archivo de texto de manera paginada, lo que significa que se muestra una pantalla a la vez. Esto es útil para visualizar archivos largos de manera más manejable.

D) **Función:** "file" es un comando utilizado para determinar el tipo de archivo de un archivo dado. No es un editor de texto, sino una herramienta de diagnóstico que proporciona información sobre el tipo de archivo, como si es un archivo de texto, una imagen, etc.

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)
   9. Acceder a el archivo iso2017-1 e ingresar Nombre y Apellido
   10. Mostrar en pantalla las últimas líneas de un archivo (tail).
2. Investigue su funcionamiento y parámetros más importantes:

(a) **shutdown:**

* Función: El comando "shutdown" se utiliza para apagar o reiniciar el sistema.
* Parámetros importantes:
  + "-r": Reinicia el sistema.
  + "-h": Apaga el sistema.
  + "-c": Cancela un apagado o reinicio programado.

(b) **reboot:**

* Función: El comando "reboot" se utiliza para reiniciar el sistema.
* Parámetros importantes: No tiene parámetros adicionales importantes.

(c) **halt:**

* Función: El comando "halt" se utiliza para apagar el sistema.
* Parámetros importantes: No tiene parámetros adicionales importantes.

(d) **locate:**

* Función: El comando "locate" se utiliza para buscar archivos en todo el sistema utilizando una base de datos indexada.
* Parámetros importantes:
  + "-i": Realiza una búsqueda insensible a mayúsculas y minúsculas.
  + "-r": Permite usar expresiones regulares en la búsqueda.

(e) **uname:**

* Función: El comando "uname" muestra información sobre el sistema operativo.
* Parámetros importantes:
  + "-a": Muestra toda la información disponible sobre el sistema.

(f) **dmesg:**

* Función: El comando "dmesg" muestra los mensajes del kernel del sistema, lo que puede ser útil para depurar problemas de hardware y sistema.
* Parámetros importantes: No tiene parámetros adicionales importantes.

(g) **lspci:**

* Función: El comando "lspci" muestra una lista detallada de todos los dispositivos PCI (Peripheral Component Interconnect) en el sistema.
* Parámetros importantes:
  + "-v": Muestra información detallada sobre los dispositivos.

(h) **at:**

* Función: El comando "at" se utiliza para programar la ejecución de comandos en un momento específico en el futuro.
* Parámetros importantes: Varios parámetros para especificar la hora y el comando a ejecutar.

(i) **netstat:**

* Función: El comando "netstat" muestra información sobre las conexiones de red, tablas de enrutamiento y estadísticas.
* Parámetros importantes:
  + "-t": Muestra conexiones TCP.
  + "-u": Muestra conexiones UDP.
  + "-n": Muestra direcciones IP y números de puerto en formato numérico.

(j) **mount:**

* Función: El comando "mount" se utiliza para montar sistemas de archivos en el sistema.
* Parámetros importantes:
  + "-t tipo\_fs": Especifica el tipo de sistema de archivos (por ejemplo, ext4, ntfs).
  + "-o opciones": Permite especificar opciones adicionales al montar.

(k) **umount:**

* Función: El comando "umount" se utiliza para desmontar sistemas de archivos previamente montados.
* Parámetros importantes: No tiene parámetros adicionales importantes.

(l) **head:**

* Función: El comando "head" muestra las primeras líneas de un archivo de texto.
* Parámetros importantes:
  + "-n N": Muestra las primeras N líneas del archivo.

(m) **losetup:**

* Función: El comando "losetup" se utiliza para configurar y administrar bucles de dispositivos de bucle (loop devices) para montar imágenes de disco.
* Parámetros importantes:
  + "-f": Encuentra un bucle de dispositivo libre.

(n) **write:**

* Función: El comando "write" permite enviar mensajes a otros usuarios conectados al sistema.
* Parámetros importantes: Debes especificar el nombre de usuario al que deseas enviar el mensaje y el mensaje en sí.

(ñ) **mkfs:**

* Función: El comando "mkfs" se utiliza para crear sistemas de archivos en particiones de disco.
* Parámetros importantes: Dependen del tipo de sistema de archivos que deseas crear (por ejemplo, "mkfs.ext4" para crear un sistema de archivos ext4).

(o) **fdisk:**

* Función: El comando "fdisk" se utiliza para administrar particiones de disco.
* Parámetros importantes: Es importante tener cuidado al usar "fdisk", ya que puede afectar las particiones y los datos en el disco. Se utilizan varios parámetros para realizar operaciones específicas en particiones, como crear, eliminar o cambiar el tipo de partición.

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

(a) **shutdown, reboot, halt:**

* Estos comandos suelen ubicarse en el directorio "/sbin/". Para ejecutarlos, es posible que necesites privilegios de superusuario (root).

(b) **locate:**

* El comando "locate" depende de una base de datos indexada para buscar archivos, y no tiene un binario específico en un directorio. El proceso "updatedb" se encarga de actualizar esta base de datos. Puedes usar "which locate" para encontrar la ubicación del comando "locate".

(c) **uname:**

* El comando "uname" suele ubicarse en el directorio "/bin/" o "/usr/bin/".

(d) **dmesg:**

* El comando "dmesg" suele ubicarse en el directorio "/bin/" o "/usr/bin/".

(e) **lspci:**

* El comando "lspci" suele ubicarse en el directorio "/usr/sbin/". Para ejecutarlo, es posible que necesites privilegios de superusuario (root).

(f) **at:**

* El comando "at" suele ubicarse en el directorio "/usr/bin/".

(g) **netstat:**

* El comando "netstat" suele ubicarse en el directorio "/bin/" o "/usr/bin/".

(h) **mount, umount:**

* Estos comandos suelen ubicarse en el directorio "/bin/" o "/usr/bin/". Para montar y desmontar sistemas de archivos, es posible que necesites privilegios de superusuario (root).

(i) **head:**

* El comando "head" suele ubicarse en el directorio "/usr/bin/".

(j) **losetup:**

* El comando "losetup" suele ubicarse en el directorio "/sbin/". Para ejecutarlo, es posible que necesites privilegios de superusuario (root).

(n) **write:**

* El comando "write" suele ubicarse en el directorio "/usr/bin/".

(ñ) **mkfs:**

* Los comandos para crear sistemas de archivos, como "mkfs.ext4", suelen ubicarse en el directorio "/sbin/". Para ejecutarlos, es posible que necesites privilegios de superusuario (root).

(o) **fdisk:**

* El comando "fdisk" suele ubicarse en el directorio "/sbin/". Para ejecutarlo y administrar particiones de disco, es necesario tener privilegios de superusuario (root).