

Trabajo Práctico 1

Introducción a los Sistemas Operativos

1.

- a. El sistema GNU es el sistema operativo similar a Unix, constituido en su totalidad por software libre. Un sistema operativo similar a Unix está constituido por muchos programas. El sistema GNU incluye todo el software GNU, además de muchos otros paquetes.
- b. Microsoft Windows:
 - Código abierto: Es un sistema operativo privativo y de código cerrado. Esto significa que los usuarios no tienen acceso a su código fuente.
 - Estabilidad y seguridad: Históricamente, ha sido más susceptible a virus y *malware* debido a su popularidad masiva en el mercado de *desktops* y a su arquitectura de permisos más laxa. Aunque ha mejorado con cada nueva versión, sigue siendo un objetivo principal para los ciberdelincuentes.
 - Flexibilidad: Está diseñado principalmente para computadoras personales (PCs), y si bien existen versiones para servidores, su uso en otros dispositivos es limitado en comparación con GNU/Linux.

Apple macOS:

- Código abierto: Es un sistema operativo de código cerrado y privativo, aunque se basa en el sistema operativo UNIX, el cual es de código abierto. Al igual que Windows, su código fuente no está disponible para el público.
- Estabilidad y seguridad: Es muy estable y seguro, en parte debido a su base UNIX y a que está diseñado para un *hardware* específico (los dispositivos de Apple), lo que le da un control estricto sobre el ecosistema. Esto lo hace un objetivo menos frecuente para los ataques de *malware* que Windows.
- Flexibilidad: Su uso está restringido a los dispositivos de Apple, por lo que su flexibilidad es muy limitada en comparación con GNU/Linux, que puede ser instalado en casi cualquier plataforma.

UNIX:

- Código abierto: Históricamente, no era de código abierto en su forma original, pero sí sentó las bases para muchos sistemas de código abierto. Hoy en día, existen versiones de código abierto, mientras que otras son privativas (como macOS).
- Estabilidad y seguridad: Es el pionero en la estabilidad y seguridad en sistemas operativos multiusuario y multitarea. Su arquitectura modular y su robusto sistema de archivos y permisos son la base de muchos sistemas operativos modernos, incluyendo GNU/Linux.

- Flexibilidad: Fue diseñado para ser portátil y escalable, lo que significa que se podía adaptar a diferentes arquitecturas de *hardware*. Esta filosofía de diseño es una de las principales herencias que GNU/Linux adoptó y perfeccionó.
- c.
- Es un Sistema Operativo tipo Unix (Unix like), pero libre
 - S.O. diseñado por miles de programadores
 - S.O. gratuito y de libre distribución (se baja desde la Web, CD, etc.)
 - Existen diversas distribuciones (customizaciones)
 - Es código abierto, lo que nos permite estudiarlo, personalizarlo, auditarlo, aprovecharnos de la documentación, etc...
- d.
- Los primeros sistemas GNU/Linux se originaron en 1992, al combinar utilidades de sistema y librerías del proyecto GNU con el núcleo Linux. Desde finales de 1990 Linux ha obtenido el apoyo de diversas empresas multinacionales del mundo de la informática, tales como IBM, Sun Microsystems, Hewlett-Packard y Novell. Actualmente GNU/Linux es comercializado en computadores de escritorio y portátiles. Si bien GNU/Linux es usado como sistema operativo en computadores de escritorio (PCs x86 y x86-64 así como Macintosh y PowerPC), computadores de bolsillo, smartphones, dispositivos empujados y otros, su mayor desarrollo se ha llevado a cabo en el mundo de los servidores y supercomputadores.
- e.
- La multitarea es la capacidad de un sistema operativo para ejecutar múltiples procesos de forma concurrente, o al mismo tiempo. Esto permite que el usuario pueda tener varios programas abiertos y funcionando sin que uno bloquee al otro.
- GNU/Linux hace uso de la multitarea. De hecho, es una de sus características fundamentales y una de las principales razones de su estabilidad y eficiencia.
- f.
- POSIX (del inglés Portable Operating System Interface) es un conjunto de estándares de interfaz de programación de aplicaciones (API) que se basa en los sistemas operativos tipo UNIX. El objetivo principal de POSIX es garantizar la portabilidad del *software*. Esto significa que un programa escrito para un sistema que cumple con estos estándares puede ser compilado y ejecutado en otro sistema compatible sin la necesidad de reescribir su código.

2.

- a.
- Una distribución Linux o distribución GNU/Linux (abreviada con frecuencia distro) es un conjunto de aplicaciones reunidas que permiten brindar mejoras para instalar fácilmente un sistema operativo basado en GNU/Linux.
- b.
- Se diferencian entre sí por las herramientas para configuración y sistemas de administración de paquetes de software para instalar. La elección de una distribución depende de las necesidades del usuario y de gustos personales.
- c.
- Debian es un sistema operativo (SO) libre para el ordenador. Debian GNU/Linux proporciona mucho más que un simple SO: incluye un amplio

rango de programas. Concretamente ofrece más de 118.000 paquetes precompilados distribuidos en un formato que hace más sencilla la instalación en un ordenador.

Orígenes (1993-1996)

Agosto de 1993: Ian Murdock funda el proyecto Debian como una distribución de Linux abierta y comunitaria.

1994-1995: Se lanzan las primeras versiones 0.9x.

Junio de 1996: Se lanza la primera versión estable, Debian 1.1 "Buzz".

Desarrollo y evolución de versiones (1996-Presente)

1996-1999: Se lanzan otras versiones tempranas: 1.2 "Rex", 1.3 "Bo", 2.0 "Hamm" (la primera multiplataforma), 2.1 "Slink", y 2.2 "Potato".

Julio de 2002: Se lanza Debian 3.0 "Woody".

Junio de 2005: Se lanza Debian 3.1 "Sarge".

Continuación: Las versiones posteriores siguen nombres de personajes de Toy Story, como 4.0 "Etch", 5.0 "Lenny", 6.0 "Squeeze", 7 "Wheezy", 8 "Jessie", 9 "Stretch", y 10 "Buster".

Debian 12 "Bookworm": La última versión estable oficial, lanzada para ofrecer una base de software confiable y segura.

<< El proyecto Debian es una asociación de personas que comparten un objetivo: queremos crear un sistema operativo libre, disponible para todo el mundo. Ahora bien, cuando utilizamos el término «libre» no estamos hablando de dinero, sino que nos referimos a la libertad del software (N. del T.: en el original en inglés se utiliza el término «free», que es sinónimo de «gratuito» y de «libre»). >>

3.

a. Cuando hablamos de un SO GNU hacemos referencia a tres elementos fundamentales:

- El kernel (núcleo)
- El Shell (intérprete de comandos)
- El FileSystem (sistema de archivos)

b. 1. El Kernel (Linux)

El kernel, o núcleo del sistema, es el corazón de GNU/Linux. Es la parte del sistema que se comunica directamente con el hardware de la computadora. El kernel es responsable de gestionar los recursos de la máquina, como la memoria, la CPU y los dispositivos de entrada/salida. Es el encargado de la multitarea y de asignar los recursos del sistema a los diferentes procesos que se están ejecutando.

2. El Shell

El shell es el intérprete de comandos. Actúa como una interfaz entre el usuario y el kernel. Cuando escribes un comando, el shell lo traduce a un

formato que el kernel puede entender para que lo ejecute. Hay varios shells, como Bash (Bourne Again Shell), que es el más popular en las distribuciones de GNU/Linux.

3. Las Utilidades GNU

Las utilidades GNU son un conjunto de herramientas de software de libre distribución. Incluyen programas básicos para la gestión de archivos y directorios, la edición de texto y la programación. Ejemplos de estas utilidades son ls (listar archivos), cp (copiar), mv (mover) y grep (buscar patrones en archivos). Estas herramientas son lo que distingue a GNU/Linux de otros sistemas operativos basados solo en el kernel de Linux.

4.

- a.
 - Ejecuta programas y gestiona dispositivos de hardware
 - Es el encargado de que el software y el hardware puedan trabajar juntos
 - Sus funciones más importantes son la administración de memoria, CPU y la E/S
 - En si, y en un sentido estricto, es el sistema operativo
 - Es un núcleo monolítico híbrido:
 - Los drivers y código del Kernel se ejecutan en modo privilegiado
 - Lo que lo hace híbrido es la capacidad de cargar y descargar funcionalidad a través de módulos
 - Está licenciado bajo la licencia GPL v2
- b. La versión actual estable del kernel de Linux es la 6.16.3, lanzada el 23 de agosto de 2025.

Antes del kernel 2.4, el esquema de versionado se definía con la siguiente estructura: MAJOR.MINOR.PATCH donde el número MINOR era clave para indicar la estabilidad de la versión.

- Si el número MINOR era impar (como 2.1, 2.3), significaba que era una versión de desarrollo. Estas versiones no estaban destinadas para uso en producción porque contenían bugs y nuevas funcionalidades aún no probadas.
- Si el número MINOR era par (como 2.2), significaba que era una versión estable. Estas eran las versiones recomendadas para los usuarios y servidores, ya que solo recibían parches de seguridad y correcciones de errores, sin nuevas funcionalidades.

La versión del kernel de Linux actualmente consta de cuatro números. Por ejemplo, asumamos que el número de la versión está compuesta de esta forma: A.B.C[.D] (ej.: 2.6.12.3).

- A: Este es el número de versión principal (Major). Cambia muy raramente y solo cuando hay cambios muy grandes en el kernel, como la introducción de nuevas arquitecturas o la eliminación de componentes importantes.

- B: Es el número de versión menor (Minor). Este número cambia con cada nueva versión del kernel. En versiones anteriores a la 2.6, este número determinaba si la rama era estable (par) o de desarrollo (impar). Hoy en día, esto ya no es relevante; simplemente indica una nueva rama principal de desarrollo.
- C: Es el número de la revisión (Revision). Este número aumenta con cada lanzamiento de parches y correcciones de errores. Cuando se corrige un fallo de seguridad o se arregla un bug, este es el número que se incrementa.
- [.D]: El cuarto número (.D) es opcional y se utiliza para indicar una corrección o un parche urgente, a menudo después de que se ha lanzado una versión oficial. Por ejemplo, una versión como 6.1.12.1 indicaría una corrección muy específica para la versión 6.1.12.

El cambio principal que se impuso a partir de la versión 2.6, y que se mantiene hasta hoy, es la eliminación del significado de "par/impar" para el número B. Esto permitió un ciclo de desarrollo más rápido y continuo.

- c. Sí, es posible. Cuando se instala un nuevo kernel en la mayoría de las distribuciones de Linux (como Ubuntu o Debian), el sistema de gestión de paquetes no elimina las versiones anteriores. Simplemente añade la nueva versión a la lista de opciones de arranque.

El gestor de arranque, como GRUB (Grand Unified Bootloader), es la herramienta que hace esto posible. Al iniciar el equipo, GRUB te presenta un menú donde puedes elegir qué versión del kernel quieres arrancar. Si no haces ninguna selección, GRUB suele iniciar la versión más reciente por defecto.

d.

- /boot: Este directorio es el lugar más importante. Contiene la imagen del kernel (vmlinuz) y otros archivos relacionados necesarios para el arranque del sistema, como el initrd (disco de memoria virtual inicial) y la configuración del gestor de arranque, como GRUB.
- /lib/modules: En este directorio se almacenan todos los módulos del kernel (.ko - kernel object). Los módulos son como *drivers* o extensiones que el kernel puede cargar o descargar dinámicamente según sea necesario, sin tener que reiniciar el sistema. Cada versión del kernel instalada tiene su propio subdirectorio dentro de /lib/modules.

5.

- a. El shell -también conocido como el intérprete de comandos línea de comandos, terminal o consola- es un programa que actúa como interfaz para comunicar al usuario con el sistema operativo mediante una ventana que espera comandos textuales ingresados por el usuario en el teclado, los interpreta y los entrega al SO para su ejecución. La respuesta del SO es mostrada al usuario en la misma ventana. A continuación, la shell queda

esperando más instrucciones. Se interactúa con la información de la manera más simple posible, sin gráficas, solo el texto.

- b. El funcionamiento del shell consiste en que, en su forma más básica, se muestra un prompt (conjunto de caracteres que se muestran en una línea de comandos para indicarnos que está a la espera de ordenes. En el Bourne Shell y sus derivados, el prompt suele ser el carácter \$ para los usuarios y # para el administrador), en donde el usuario teclea una orden en el teclado y finaliza la orden (normalmente con la tecla Intro/Enter), y la computadora ejecuta la orden, proporcionando una salida de texto.

- c. /bin/sh - Bourne Shell: Está disponible en todas las versiones de UNIX y es lo suficientemente básico como para que funcione en todas las plataformas.

/bin/bash - Bourne Again Shell: Uno de los shells más avanzados y populares en GNU/Linux. Tiene licencia GNU. Ofrece las mismas capacidades que csh, pero incluye funciones avanzadas: un historial de los comandos ejecutados, que se conserva incluso al pasar de una sesión a otra, accesible utilizando los cursores (arriba/abajo), auto completado de nombres de comandos o archivos presionando TAB, manejo de varios tipos de redirecciones de entrada/salida.

/bin/zsh: Se diseñó para poder usarse interactivamente. Se le han incorporado muchas de las características principales de otras shells de Unix como bash, ksh y además posee características propias originales. macOS Catalina, lanzada en octubre de 2019 adoptó a Zsh como la shell predeterminada, reemplazando a Bash.

El Bourne Shell (sh) es bastante básico y, por ende, flexible entre todas las plataformas. Por otro lado, Bourne Again Shell (bash) es igualmente muy conocido pero con muchas más funciones que el Bourne Shell. Finalmente, zsh es más interactiva, adoptando bastantes características de otras shells de Unix.

- d. Los comandos internos del shell no se encuentran en un path específico porque están embebidos directamente en el intérprete de comandos, como Bash (Bourne Again Shell). No son archivos ejecutables separados. Cuando se escribe un comando como cd (cambiar de directorio), echo o pwd (imprimir directorio de trabajo), el shell lo ejecuta sin tener que buscarlo en ningún directorio.

Los comandos externos son programas ejecutables que se encuentran en directorios específicos. El shell los busca en una lista de rutas de directorios definida en la variable de entorno \$PATH. Cuando se escribe un comando como ls, grep o nano, el shell recorre los directorios listados en \$PATH hasta que encuentra el archivo ejecutable del comando.

- e. La shell no forma parte del kernel básico del SO; sino que la misma “dialoga” con el kernel. La shell es iniciada por un proceso denominado “login”, y dado que cada usuario tiene asignado una shell por defecto, la misma se inicia cada vez que un usuario comienza a trabajar en su estación de trabajo (es decir se “loguea” en una terminal). Dentro del contenido del archivo

/etc/passwd, se puede ver cual es la shell que cada usuario tiene asignada por defecto.

- f. El shell de un usuario se define en el sexto campo del archivo /etc/passwd. Cada línea en este archivo representa una cuenta de usuario y tiene la siguiente estructura, con los campos separados por dos puntos (:):

nombre_usuario:contraseña:UID:GID:GECOS:directorio_home:**shell**

No cualquier usuario puede realizar esta tarea. La modificación del archivo /etc/passwd requiere privilegios de superusuario, es decir, del usuario root. Esto se debe a que este archivo es crítico para la seguridad y la funcionalidad del sistema. Si un usuario sin privilegios pudiera cambiar su shell a un programa malicioso, podría comprometer la seguridad de la máquina.

Los administradores del sistema usan herramientas como chsh (change shell) para modificar el shell de un usuario de forma segura. Por ejemplo, para cambiar el shell del usuario juan a /bin/zsh, un administrador con permisos de sudo usaría el siguiente comando:

```
sudo chsh -s /bin/zsh juan
```

6.

- a. Un sistema de archivos es la estructura lógica que un sistema operativo utiliza para organizar, almacenar, nombrar y gestionar archivos en un dispositivo de almacenamiento (como un disco duro, SSD, o memoria USB).
- b. La estructura de directorios en GNU/Linux es un árbol jerárquico que comienza desde un directorio raíz, simbolizado con la barra /.

La sigla FHS se refiere a Filesystem Hierarchy Standard, que es un estándar que define la estructura principal de directorios y el contenido de cada uno, asegurando la consistencia entre las diferentes distribuciones de GNU/Linux.

- /: Directorio raíz. Es el directorio principal de todo el sistema de archivos. Todo lo demás se encuentra dentro de él.
- /bin: Binarios esenciales del usuario. Contiene programas ejecutables esenciales para todos los usuarios, como ls, cp, y mv.
- /sbin: Binarios del sistema. Contiene programas ejecutables para la administración del sistema que solo pueden ser ejecutados por el root o con privilegios.
- /etc: Archivos de configuración. Contiene los archivos de configuración del sistema, como /etc/passwd y /etc/hosts.
- /home: Directorios de usuario. Aquí se almacenan los directorios personales de cada usuario.
- /root: Es el directorio personal del usuario root.
- /dev: Archivos de dispositivos. Contiene archivos especiales que representan dispositivos de *hardware* (ej: /dev/sda1).

- /proc: Archivos del sistema en ejecución. Es un *pseudo* sistema de archivos que contiene información sobre los procesos del sistema y el kernel.
 - /usr: Jerarquía de solo lectura del usuario. Contiene la mayoría de los programas y librerías que no son esenciales para el arranque.
 - /var: Datos variables. Contiene archivos que cambian constantemente, como *logs* del sistema (/var/log).
- c. GNU/Linux soporta una gran variedad de sistemas de archivos, tanto nativos como de otros sistemas operativos. Algunos de los más importantes son:
- ext2, ext3, ext4: Nativos de Linux. ext4 es el más común en la actualidad, ya que es la evolución de los anteriores y ofrece mejoras significativas en rendimiento y robustez.
 - XFS: Creado por SGI, conocido por su alto rendimiento.
 - Btrfs: Un sistema de archivos moderno con características como snapshots y checksums.
 - ZFS: Otro sistema avanzado con funcionalidades como gestión de volúmenes, snapshots y duplicación de datos.
 - FAT/FAT32, exFAT: De uso común en memorias USB y tarjetas SD.
 - NTFS: El sistema de archivos nativo de Windows.
 - HFS+: El sistema de archivos de Apple macOS.
- d. Sí, es posible. GNU/Linux tiene soporte incorporado para leer y escribir en particiones FAT y NTFS. Esto se logra a través de controladores (drivers) y utilidades. Para NTFS, se utiliza el controlador NTFS-3G, que permite un soporte completo de lectura y escritura.

7.

- a. Una partición es una división lógica de un disco duro. El sistema operativo ve cada partición como una unidad de almacenamiento separada.

Tipos:

- Primaria: Un disco puede tener hasta cuatro particiones primarias.
- Extendida: Es un tipo especial de partición primaria que actúa como un contenedor para las particiones lógicas. Solo puede haber una partición extendida por disco.
- Lógica: Particiones contenidas dentro de una partición extendida. El número de particiones lógicas es prácticamente ilimitado.

Ventajas:

- Organización: Permite separar los datos del sistema operativo de los datos del usuario.
- Seguridad: En caso de que una partición se corrompa, los datos en otras particiones no se verán afectados.

- Multisistema: Permite instalar varios sistemas operativos en el mismo disco duro (*dual-booting*).

Desventajas:

- Ineficiencia: El espacio no utilizado en una partición no puede ser usado por otra partición sin herramientas de redimensionamiento.
- Complejidad: Un mal particionamiento puede llevar a problemas de espacio o estabilidad.

b. En GNU/Linux, las particiones se identifican por el nombre del disco y el número de partición, en el directorio /dev.

- Discos IDE: Se nombran como hda, hdb, etc., y las particiones como hda1, hda2.
- Discos SCSI, SATA, USB, NVMe: Se nombran como sda, sdb, etc. Las particiones se nombran con un número:
 - sda1: Primera partición del primer disco SATA.
 - sdb2: Segunda partición del segundo disco SATA.
 - Las particiones lógicas se numeran a partir del 5 (ej: sda5, sda6).

c. Cómo mínimo se necesita una partición para el /.

1. Partición Raíz:

- Tipo: Primaria
- Identificación: sda1
- Tipo de FS: ext4
- Punto de Montaje: /

2. Partición de Intercambio (Swap):

- Tipo: Primaria o lógica
- Identificación: sda2
- Tipo de FS: swap
- Punto de Montaje: Ninguno (se usa para memoria virtual)

d. Uso en *Desktop*:

- / (raíz): ext4, 20-30 GB
- /home: ext4, el resto del disco
- swap: swap, igual o el doble de la memoria RAM.

Uso en Servidor:

- / (raíz): ext4, 10-20 GB
- /var: ext4, 20-50 GB (para *logs*, bases de datos, web)
- /home: ext4, 1-2 GB (para usuarios)
- /tmp: ext4, 5-10 GB

- swap: swap, 4 GB

Particionamiento de SSD:

- / (raíz): ext4
- No usar swap o usar un archivo swap en lugar de una partición para reducir la escritura en la unidad.

- e. GNU Parted: Herramienta de línea de comandos, potente y flexible. Se usa en entornos de servidor.

GParted: La versión gráfica de GNU Parted. Es muy intuitiva y fácil de usar, ideal para usuarios de *desktop*.

Disk Utility (Ubuntu): Herramienta gráfica incorporada en la mayoría de las distribuciones de Linux.

fdisk, sfdisk, cfdisk: Utilidades de línea de comandos más antiguas, aún muy usadas por su sencillez y eficacia.

8.

- a. El BIOS (Basic Input Output System) es un software de bajo nivel que se halla en el motherboard. Cuando se arranca la computadora el BIOS se ejecuta, realizando el POST (Power-on self-test), que incluye rutinas que, entre otras actividades, fijan valores de las señales internas, y ejecutan test internos (RAM, el teclado, y otros dispositivos a través de los buses ISA y PCI).

Luego se lee el primer sector del disco de inicio (seleccionado de entre un conjunto de posibles dispositivos de arranque), llamado MBR (master boot record). Esto se carga en memoria y se ejecuta. El MBR puede contener un código de arranque denominado MBC (master boot code) y una marca de 2 bytes que indica su presencia o puede solamente contener la tabla de particiones. En el último caso el BIOS ignora este MBR.

- b. UEFI está definido para las arquitecturas IA32, IA64, AMD64 y ARM, pero para poder hacer una comparación objetiva con el BIOS tradicional detallaremos solamente el proceso de arranque de UEFI en las arquitecturas IA32 y AMD64. La interfaz de comunicación entre el preboot environment y el sistema operativo, que define UEFI, consta simplemente de tablas de datos en memoria con información sobre el hardware y configuración del firmware. Estas estructuras tienen, además, punteros a las rutinas que implementan los servicios que el firmware ofrece a los bootloaders y a otras aplicaciones UEFI. UEFI provee un Boot Manager que permite cargar aplicaciones y drivers UEFI desde el UEFI filesystem (este filesystem puede ser FAT12, FAT16 o FAT32) o por red. En este esquema el boot loader (por ejemplo la versión de Grub2 para UEFI) es un tipo especial de aplicación UEFI que al ser cargado se ejecuta ya en “modo protegido” o en “long mode” dependiendo de la arquitectura y tiene a su disposición los servicios de UEFI. De esta forma el boot loader no tiene las restricciones de tamaño impuestas por el antiguo BIOS ni la imposición de tener que arrancar en “modo real” (con direcciones de 20 bits).

- c. El MBR (Master Boot Record) es el primer sector de un disco duro o dispositivo de almacenamiento. Contiene la tabla de particiones del disco y un pequeño programa de arranque llamado MBC (Master Boot Code). El MBC es el código ejecutable dentro del MBR. Su tarea es encontrar la partición de arranque, cargar el gestor de arranque de esa partición y pasarle el control para que inicie el sistema operativo.
- d. GPT (GUID Partition Table) especifica la ubicación y formato de la tabla de particiones en un disco duro. Es parte de EFI. Puede verse como una sustitución del MBR como era pensado en la BIOS. GPT usa modo de direccionamiento lógico (LBA, logical block addressing) en vez del modo cilindro-cabeza-sector usado con el MBR. El MBR “heredado” se almacena en el LBA 0.
- e. Un Gestor de Arranque (*bootloader*) es un programa que se carga en la memoria después del BIOS o UEFI y antes que el sistema operativo. Su funcionalidad principal es presentar al usuario un menú con los sistemas operativos instalados en la máquina y cargar el seleccionado.
 - Tipos de gestores de arranque conocidos:
 - GRUB2 (Grand Unified Bootloader): El gestor de arranque más popular en GNU/Linux.
 - LILO (Linux Loader): Un gestor de arranque más antiguo.
 - Systemd-boot: Gestor de arranque simple y ligero usado en algunas distribuciones.
 - Ubicación: Generalmente, se instala en el MBR o en la partición EFI (para sistemas UEFI) del disco duro.
- f.
 - i. Encendido: El usuario enciende la computadora.
 - ii. POST: El BIOS/UEFI ejecuta el POST para verificar el *hardware*.
 - iii. Carga del Gestor de Arranque: El BIOS/UEFI busca un dispositivo de arranque (disco duro, USB) y carga el gestor de arranque (ej: GRUB) en la memoria RAM.
 - iv. Menú del Gestor de Arranque: El gestor de arranque muestra un menú con los sistemas operativos disponibles.
 - v. Carga del SO: El usuario selecciona un sistema operativo, y el gestor de arranque carga el kernel y los archivos necesarios en la memoria RAM.
 - vi. Inicio del SO: El kernel toma el control y completa el proceso de arranque del sistema operativo.
- g.
 - i. BIOS/UEFI y GRUB: El BIOS/UEFI carga GRUB, que muestra un menú con las opciones de arranque.

- ii. Carga del Kernel e initrd: GRUB carga la imagen del kernel (vmlinuz) y un disco de memoria virtual inicial (initrd.img) desde el directorio /boot a la memoria RAM.
 - iii. Kernel toma el control: El kernel se inicializa, configura el *hardware* y monta el sistema de archivos raíz (/) usando los controladores en initrd.
 - iv. systemd: El kernel inicia el proceso init (que en la mayoría de las distribuciones modernas es systemd).
 - v. Servicios de arranque: systemd lee sus configuraciones y arranca los servicios del sistema (red, bases de datos, etc.).
 - vi. Login: Una vez que todos los servicios necesarios han iniciado, el sistema presenta la pantalla de inicio de sesión o la interfaz gráfica del usuario.
- h.
- i. Comando de shutdown: Un usuario con privilegios de administrador ejecuta un comando como shutdown, halt o reboot.
 - ii. Envío de señales SIGTERM: El sistema envía una señal (SIGTERM) a todos los procesos en ejecución, pidiéndoles que se cierren de manera ordenada.
 - iii. Espera de finalización: El sistema espera un breve período de tiempo para que los procesos se cierren. Si un proceso no responde, se le envía una señal SIGKILL para terminarlo de forma forzada.
 - iv. Sincronización de datos: El sistema sincroniza los datos en la memoria con el disco duro para evitar la pérdida de información.
 - v. Desmontaje del File System: El sistema de archivos se desmonta en modo de solo lectura.
 - vi. Apagado o Reinicio: Finalmente, el sistema operativo le dice al *hardware* que se apague o se reinicie.
- i. Sí, es posible. Esta configuración se conoce como arranque dual o dual-boot. Un solo disco duro puede dividirse en múltiples particiones. Es posible instalar un sistema operativo en una partición (ej: Windows) y otro en una partición diferente (ej: GNU/Linux). Durante el proceso de instalación de Linux, el gestor de arranque (como GRUB) detecta la presencia de Windows y se configura para ofrecer un menú de selección al iniciar la computadora, permitiendo al usuario elegir qué sistema operativo desea cargar.
- 9.
- a. Los archivos en GNU/Linux se identifican por su nombre y su ubicación en el sistema de archivos. A diferencia de Windows, la extensión del archivo (ej: .exe) no define el tipo de archivo de forma estricta, sino que es una convención. Los permisos, el propietario y el grupo de un archivo también son parte de su identificación.

- b. Vim: Es un editor de texto muy potente, pero con una curva de aprendizaje pronunciada. Funciona en modos:

- Normal: Para navegar por el archivo.
- Insert: Para escribir.
- Visual: Para seleccionar texto.
- Comando: Para ejecutar comandos de edición.

Nano: Editor de texto simple y amigable para principiantes. Las opciones de comandos se muestran en la parte inferior de la pantalla. Es ideal para ediciones rápidas.

Mcedit: Editor del gestor de archivos Midnight Commander. Ofrece una interfaz similar a otros editores visuales, con menús y atajos de teclado fáciles de recordar.

cat: Muestra el contenido completo de un archivo en la terminal. Útil para archivos pequeños.

more: Muestra el contenido de un archivo página por página.

less: Una versión más avanzada de more. Permite navegar hacia adelante y hacia atrás en el archivo.

- c. IntroALosSistemasOperativos/2025/Prácticas/Práctica 1/prueba.exe

d.

e.

- i. cd: accede a determinada ruta desde donde se encuentra uno parado;
- ii. mkdir: crea un directorio;
- iii. rmdir: elimina un directorio;
- iv. ln: crea enlaces a archivos (como el acceso directo en Windows);
- v. tail: muestra las últimas líneas de un archivo;
- vi. locate: busca archivos por nombre;
- vii. ls: lista directorios y archivos;
- viii. pwd: muestra el directorio actual;
- ix. cp: copia archivos y directorios;
- x. mv: mueve o renombra archivos y directorios;
- xi. find: busca archivos recursivamente.

10.

- a. mkdir ISOCSO
- b. cd ISOCSO/
- c. touch isocso.txt
touch isocso.csv
- d. ls

- e. pwd
- f. find . -name "iso*"
- g. df -h
- h. who
- i. vim isocso.txt
 - i
 - escribo Federico Dobal-
 - Escape
 - :wq
- j. tail isocso.txt

11.

Comando	Funcionamiento	Parámetros	Ubicación
man	Interfaz para los manuales de referencia del sistema	man [man opciones] [[sección] página ...] ... man -k [apropos opciones] regexp ... man -K [man opciones] [sección] término ... man -f [whatis opciones] página ... man -l [man opciones] archivo ... man -w -W [man opciones] página ...	/usr/local/etc/man_db.conf archivo de configuración man-db. /usr/share/man jerarquía de páginas de manual global.
shutdown	Detener, apagar o reiniciar la máquina	shutdown [OPTIONS...] [TIME] [WALL...]	/sbin/shutdown
reboot	Reiniciar el equipo	reboot [OPCIONES...]	/sbin/reboot
halt	Detiene el equipo	halt [OPCIONES...]	/sbin/halt
uname	Imprime información del sistema	uname [OPCIONES...]	/usr/bin/uname
dmesg	Imprime o controla el buffer del anillo del kernel	dmesg [OPCIONES...] dmesg --clear dmesg --read-clear [OPCIONES...] dmesg --console-level level dmesg --console-on dmesg --console-off	/usr/bin/dmesg
lspci	Listar todos los dispositivos PCI	lspci [OPCIONES...]	/usr/share/misc/pci.ids /usr/share/misc/ <u>pci.ids</u>

			<code>.gz</code> <code>\$XDG_CACHE_HOME/pci-ids</code>
at	Ejecuta órdenes a una determinada hora	<code>at [-V] [-q cola] [-f archivo] [-u nombreusuario] [-mMlv] timespec ..."</code> <code>at [-V] [-q cola] [-f archivo] [-u nombreusuario] [-mMkv] [-t tiempo]</code> <code>at -c tarea [...]</code> <code>at [-V] -l [-o formatotiempo] [tarea ...]</code> <code>at [-V] [-q cola] [-o formatotiempo] [tarea ...]</code> <code>at [-rd] trabajo [...]</code>	<code>/var/spool/cron/atjobs</code> <code>/var/spool/cron/atspool</code> <code>/proc/loadavg</code> <code>/var/run/utmp</code> <code>/etc/at.allow</code> <code>/etc/at.deny</code>
netstat	Muestra conexiones de red, tablas de enrutamiento, estadísticas de interfaces, conexiones enmascaradas e información de los miembros de grupos de multidifusión.	<code>netstat [opciones_familia_dirección] [--tcp -t] [--udp -u] [--udplite -U] [--sctp -S] [--raw -w] [--l2cap -2] [--rfcomm -f] [--listening -l] [--all -a]</code> <code> [--numeric -n] [--numeric-hosts]</code> <code> [--numeric-ports] [--numeric-users]</code> <code> [--symbolic -N] [--extend -e[--extend -e]]</code> <code> [--timers -o] [--program -p] [--verbose -v]</code> <code> [--continuous -c] [--wide -W]</code> <code>netstat {--route -r}</code> <code>[opciones_familia_dirección]</code> <code> [--extend -e[--extend -e]] [--verbose -v]</code> <code> [--numeric -n] [--numeric-hosts] [--numeric-ports]</code> <code> [--numeric-users]</code> <code> [--continuous -c]</code> <code>netstat {--interfaces -i} [--all -a]</code> <code> [--extend -e[--extend -e]] [--verbose -v]</code> <code> [--program -p] [--numeric -n] [--numeric-hosts]</code> <code> [--numeric-ports]</code> <code> [--numeric-users] [--continuous -c]</code> <code>netstat {--groups -g} [--numeric -n]</code> <code> [--numeric-hosts] [--numeric-ports]</code> <code> [--numeric-users] [--continuous -c]</code> <code>netstat {--masquerade -M} [--extend -e]</code> <code> [--numeric -n] [--numeric-hosts] [--numeric-ports]</code> <code> [--numeric-users] [--continuous -c]</code> <code>netstat {--statistics -s} [--tcp -t] [--udp -u]</code> <code> [--udplite -U] [--sctp -S] [--raw -w]</code> <code>netstat {--version -V}</code> <code>netstat {--help -h}</code>	<code>/etc/services</code> <code>/proc</code> <code>/proc/net/...</code> <code>/sys/kernel/debug/bluetooth/...</code>
head	Muestra la primera	<code>head [OPCIONES...] [ARCHIVO...]</code>	<code>usr/bin/head</code>

	parte de un archivo		
tail	Muestra la parte final de un archivo	tail [OPCIONES...] [ARCHIVO...]	

12.

- a. En Linux, un proceso es una instancia de un programa en ejecución, con su propio ID (PID) único y un espacio de memoria. Se gestionan mediante herramientas como ps, top o htop, que muestran su estado (ejecutándose, detenido, etc.), consumo de recursos y jerarquía. Los procesos pueden ser del sistema o de usuario, y sus estados pueden incluir activo, dormido, detenido o zombi, con la posibilidad de finalizarlos con el comando kill.

PID significa Process ID o Identificador de Proceso. Es un número entero que el sistema operativo asigna de forma única a cada programa o tarea en ejecución (proceso).

PPID significa Process Parent ID (ID del Proceso Padre). Es el ID del proceso (PID) del proceso que creó o inició otro proceso. El PPID establece una relación padre-hijo entre procesos, formando una estructura de árbol, donde cada proceso hijo tiene el mismo PPID que su proceso padre.

Todos los procesos tienen un PID (Process ID) y un PPID (Parent Process ID), con la excepción del proceso "PID 0", un pseudo-proceso que inicia al sistema y no tiene padre, y el proceso "PID 1" (init/systemd), que se considera la raíz de toda la jerarquía de procesos y, por lo tanto, tiene un PPID de 0.

Atributos más importantes de los procesos:

- PID (ID de Proceso): Un número único que identifica a cada proceso en el sistema.
- Nombre del Proceso: El nombre del comando que se ejecutó para iniciar el proceso.
- Usuario Propietario: El nombre del usuario que creó o es dueño del proceso.
- Estado del Proceso: Indica en qué fase del ciclo de vida se encuentra el proceso, como:
 - En ejecución: El proceso está actualmente utilizando la CPU.
 - En suspensión (interrumpible e ininterrumpible): El proceso ha sido pausado y está esperando un evento o recursos.
 - Detenido: El proceso ha sido detenido por una señal, como SIGSTOP, y puede reanudarse con SIGCONT.
 - Zombi: El proceso ha terminado, pero su entrada en la tabla de procesos todavía existe hasta que su proceso padre la recolecta.

- Padre/Hijo: Los procesos existen en una jerarquía. Cada proceso tiene un proceso padre y puede crear procesos hijo, a quienes puede gestionar.

b.

Comando	Funcionamiento	Parámetros	Ubicación
top	Muestra los procesos de Linux	top [OPCIONES...]	/usr/bin/top
htop	Es un visor de procesos multiplataforma. Además de lo que hace top, permite scrollar horizontal y verticalmente, e interactuar con el mouse	htop [-dCFhpustvH] pcp-htop [-dCFhpustvH] [--host/-h host]	/usr/bin/htop
ps	Muestra información de los procesos activos actualmente	ps [OPCIONES...]	/usr/bin/ps
pstree	Muestra un árbol de los procesos activos actualmente	pstree [-a, --arguments] [-c, --compact-not] [-C, --color attr] [-g, --show-pgids] [-h, --highlight-all, -H pid, --highlight-pid pid] [-l, --long] [-n, --nu- meric-sort] [-N, --ns-sort ns] [-p, --show-pids] [-s, --show-parents] [-S, --ns-changes] [-t, --thread-names] [-T, --hide-threads] [-u, --uid-changes] [-Z, --security-context] [-A, --ascii, -G, --vt100, -U, --unicode] [pid, user] pstree -V, --version	/usr/bin/pstree
kill	Envía una señal a un proceso. La señal por defecto es TERM (terminar proceso).	kill [OPCIONES...] <pid> [...]	/usr/bin/kill
pgrep	Examina los procesos que se están ejecutando actualmente y enumera los ID de procesos que coinciden con los criterios de selección indicados en el patrón.	pgrep [OPCIONES...] pattern	/usr/bin/pgrep
pkill	Termina los procesos que se están ejecutando actualmente y enumera los ID de procesos que coinciden con	pkill [OPCIONES...] pattern	/usr/bin/pkill

	los criterios de selección indicados en el patrón.		
killall	Termina procesos por nombre.	killall [-Z, --context pattern] [-e, --exact] [-g, --process-group] [-i, --interactive] [-n, --ns PID] [-o, --older-than TIME] [-q, --quiet] [-r, --regexp] [-s, --signal SIGNAL, -SIGNAL] [-u, --user user] [-v, --verbose] [-w, --wait] [-y, --younger-than TIME] [-I, --ignore-case] [-V, --version] [--] name ... killall -l, --list killall -V, --version	/usr/bin/killall
renice	Altera la prioridad de los procesos actuales.	renice [--priority --relative] priority [-g -p -u] identifier...	/usr/bin/renice
xkill	Es una utilidad para forzar al servidor X a cerrar las conexiones con los clientes.	xkill [-display displayname] [-id resource] [-button number] [-frame] [-all] [-version]	/usr/bin/xkill
atop	Monitor avanzado de sistemas y procesos.	Live measurement in bar graph mode: atop -B[H] [-t [absdir]] [interval [samples]] Live measurement cgroups in text mode: atop -G [-t [absdir]] [-2 -3 -4 -5 -6 -7 -8 -9] [-a] [-C -M -D -A] [interval [samples]] Live measurement processes in text mode: atop [-t [absdir]] [-g -m -d -n -u -p -s -c -v -o -y -Y] [-C -M -D -N -A] [-fX1xR] [interval	/usr/bin/atop

		<p>[samples]]</p> <p>Live generation of parsable output (white-space separated or JSON):</p> <p>atop [-Plabel[,label]... [-Z]] [-Jlabel[,label]...] [interval [samples]]</p> <p>Write raw log files:</p> <p>atop -w rawfile [-a] [-S] [interval [samples]]</p> <p>Analyze raw log files in bar graph mode:</p> <p>atop -B[H] -r [rawfile yyy...] [-b [YYYYMMDD]hhmm[ss]] [-e [YYYYMMDD]hhmm[ss]]</p> <p>Analyze cgroups from raw log files in text mode:</p> <p>atop -G [-2 -3 -4 -5 -6 -7 -8 -9] [-a] -r [rawfile yyy...] [-b [YYYYMMDD]hhmm[ss]] [-e [YYYYMMDD]hhmm[ss]]</p> <p>Analyze processes from raw log files in text mode:</p> <p>atop -r [rawfile yyy...] [-b [YYYYMMDD]hhmm[ss]] [-e [YYYYMMDD]hhmm[ss]] [-g -m -d -n -u -p -s -c -v -o -y -Y] [-C -M -D -N -A] [-fFX1xR]</p> <p>Generate parsable output from raw log files (white-space separated or JSON):</p> <p>atop -r [rawfile yyy...] [-b [YYYYMMDD]hhmm[ss]] [-e</p>	
--	--	---	--

		[YYYYMMDD]hhmm[ss] [-Plabel[,label]... [-Z]] [-Jlabel[,label]...]	
nice	Ejecuta un programa con la prioridad de planificación modificada.	nice [OPCIONES...] [ORDEN [ARG]...]	/usr/bin/nice

13.

a.

- i. **BIOS/UEFI y POST:** Al encender la PC, el *firmware* (BIOS o UEFI) realiza un **Power-On Self-Test** (POST) para verificar el *hardware*.
 - ii. **Carga del gestor de arranque:** El BIOS/UEFI busca un dispositivo de arranque (como el disco duro) y carga el **gestor de arranque** (ej: GRUB) en la memoria.
 - iii. **Carga del Kernel:** El gestor de arranque carga la imagen del **kernel** (vmlinuz) y el **initrd** (disco de memoria virtual inicial) desde la partición de arranque hacia la memoria RAM.
 - iv. **Inicio de init:** El kernel se inicializa, configura el *hardware*, y ejecuta el primer proceso del sistema, llamado **init**. Este proceso siempre tiene el ID de proceso (PID) 1.
 - v. **Ejecución de Runlevels:** init lee el archivo de configuración */etc/inittab* para determinar el nivel de ejecución (o *runlevel*) predeterminado y ejecutar los *scripts* correspondientes.
 - vi. **Inicio de servicios y login:** Se inician los servicios del sistema, se montan los sistemas de archivos y se presenta la pantalla de inicio de sesión (*login*).
- b. El kernel ejecuta el proceso init. Principalmente, el objetivo de init es ser el padre de todos los procesos posteriores. Gestiona el inicio y la detención de los servicios, supervisa los procesos huérfanos (procesos cuyos padres han terminado) y, en esencia, mantiene el sistema en funcionamiento.
- c. Los niveles de ejecución (runlevels) son modos de operación del sistema que definen qué servicios se inician o se detienen. Un runlevel especifica un conjunto particular de servicios y funcionalidades. Por ejemplo, un runlevel podría iniciar todos los servicios para un entorno gráfico, mientras que otro solo iniciaría los servicios esenciales para la línea de comandos.
- El objetivo de los *runlevels* es permitir que el administrador del sistema inicie el sistema en un estado específico de funcionalidad. Esto es útil para tareas de mantenimiento (como el modo de usuario único), o para arrancar el sistema con servicios mínimos para solucionar problemas.
- d.
- i. **Runlevel 0: Apagado (Halt).** El sistema se apaga.

- ii. **Runlevel 1 (S o s): Modo de usuario único.** Se usa para tareas de mantenimiento, sin red y con servicios mínimos.
 - iii. **Runlevel 2: Multiusuario sin red.** Es un modo multiusuario estándar.
 - iv. **Runlevel 3: Multiusuario con red.** Es el modo completo de línea de comandos.
 - v. **Runlevel 4: Reservado.** No tiene una función estándar y puede ser configurado por el usuario.
 - vi. **Runlevel 5: Multiusuario con red y entorno gráfico.** Es el modo de escritorio completo.
 - vii. **Runlevel 6: Reinicio (Reboot).** El sistema se reinicia.
- e. La finalidad de `/etc/inittab` es definir el **nivel de ejecución predeterminado** y qué procesos deben ser ejecutados en cada *runlevel*.

Almacena una lista de procesos que `init` debe iniciar o detener cuando el sistema cambia de *runlevel*.

Cada línea del archivo tiene la siguiente estructura:

`id:runlevels:action:process`

- **id:** Identificador único de 1 a 4 caracteres.
 - **runlevels:** Nivel o niveles de ejecución en los que se debe ejecutar el proceso.
 - **action:** La acción que `init` debe tomar (ej: `initdefault` para definir el *runlevel* por defecto, `sysinit` para ejecutar *scripts* de inicialización, o `respawn` para reiniciar un proceso si se detiene).
 - **process:** El comando o *script* a ejecutar.
- f. Para cambiar al *runlevel* <Y> se debe usar el comando `init Y`.
- No es un cambio permanente. Solo surte efecto hasta el siguiente reinicio del sistema. Al reiniciar, el sistema volverá a leer el archivo `/etc/inittab` y arrancará en el *runlevel* que se haya definido como predeterminado. Para hacer un cambio permanente, debes editar el archivo `/etc/inittab` y modificar la línea de `initdefault`.
- g. Los *scripts* **RC** (Runlevel Control) son *scripts* de *shell* que se ejecutan para iniciar o detener servicios cuando el sistema entra o sale de un *runlevel*. Por ejemplo, un *script* RC podría iniciar el servidor web o detener la base de datos.

Se almacenan en directorios como `/etc/rc.d` o `/etc/init.d`.

El sistema determina qué *script* ejecutar basándose en los **enlaces simbólicos** que se encuentran en directorios específicos para cada *runlevel* (ej: `/etc/rc.d/rc5.d` para el *runlevel* 5).

- **S:** Los *scripts* que comienzan con S (del inglés Start) se ejecutan cuando se ingresa a un *runlevel*.
- **K:** Los *scripts* que comienzan con K (del inglés Kill) se ejecutan cuando se sale de un *runlevel* para detener los servicios.

14.

- a. Es un sistema de gestión del sistema y de servicios para GNU/Linux que actúa como el sistema de inicio (PID 1), gestionando el arranque del sistema operativo, los servicios, los puntos de montaje, el registro de eventos y la configuración de los componentes del sistema.
- b. Una **Unit** (unidad) en **systemd** es la unidad básica de control. Es un archivo de configuración que describe un recurso o servicio que **systemd** puede gestionar. Existen varios tipos de unidades, como:
 - **Service unit (.service)**: para gestionar servicios de *daemon* (ej: un servidor web).
 - **Target unit (.target)**: para agrupar otras unidades y definir un estado del sistema (similar a un *runlevel*).
 - **Mount unit (.mount)**: para gestionar puntos de montaje de sistemas de archivos.
 - **Device unit (.device)**: para gestionar dispositivos de *hardware*.

c.