Trabajo Práctico 1

Introducción a los Sistemas Operativos

1.
    1. 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.
    2. 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.

3. • 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…
4. 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 empotrados y otros, su mayor desarrollo se ha llevado a cabo en el mundo de los servidores y supercomputadores.
5. 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.
6. 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.

1. 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.
2. 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.
3. 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.

1. 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)

2. 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.

1. • 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

2. 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.

3. 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.

4.

- /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.

1. 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.

2. 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.

3. /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.
4. 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.
5. 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.
6. 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.

1. 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).
2. 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).
3. 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.

4. 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.

1. 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.

2. 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).

3. 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)

4. 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.

5. 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.

1. 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.

2. 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).
3. 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.
4. 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.
5. 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.
6.

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.

7.

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.

8.

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.

9. 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.

1. 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.

2. 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.

3. IntroALosSistemasOperativos/2025/Prácticas/Práctica 1/prueba.exe

4.

5.

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.

1. mkdir ISOCSO
2. cd ISOCSO/
3. touch isocso.txt
   touch isocso.csv
4. ls
5. pwd
6. find . -name "iso*"
7. df -h
8. who
9. vim isocso.txt
   i
   -escribo Federico Dobal-
   Escape
   :wq
10. 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	dmesg [OPCIONES…] dmesg --clear	/usr/bin/dmesg

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

		netstat {--version|-V} netstat {--help|-h}	
head	Muestra la primera parte de un archivo	head [OPCIONES…] [ARCHIVO…]	usr/bin/head
tail	Muestra la parte final de un archivo	tail [OPCIONES…] [ARCHIVO…]	

12.

1. 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.

2.

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 scrollear 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 los criterios de selección indicados en el patrón.	pkill [OPCIONES…] pattern	/usr/bin/pkill
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]	/usr/bin/atop

		[-fFX1xR] [interval [samples]] Live generation of parsable output (white-space separated or JSON): atop [-Plabel[,label]... [-Z]] [-Jlabel[,label]...] [interval [samples]] Write raw log files: atop -w rawfile [-a] [-S] [interval [samples]] Analyze raw log files in bar graph mode: atop -B[H] -r [rawfile|yyy...] [-b [YYYYMMDD]hhmm[ss]] [-e [YYYYMMDD]hhmm[ss]] Analyze cgroups from raw log files in text mode: atop -G [-2|-3|-4|-5|-6|-7|-8|-9] [-a] -r [rawfile|yyy...] [-b [YYYYMMDD]hhmm[ss]] [-e [YYYYMMDD]hhmm[ss]] Analyze processes from raw log files in text mode: 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] Generate parsable output from raw log files (white-space separated or JSON): atop -r [rawfile|yyy...] [-b [YYYYMMDD]hhmm[ss]] [-e [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.

1.

i. **BIOS/UEFI y POST**: Al encender la PC, el *firmware* (BIOS o UEFI) realiza un **P**ower-**O**n **S**elf-**T**est (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*).

2. 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.

3. 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.

4.

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.

5. 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.

6. 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.

7. 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.

1. 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.

2. 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*.
3. El comando systemctl se usa en Linux para controlar el sistema de inicio systemd y administrar los servicios del sistema. Permite a los usuarios iniciar, detener, reiniciar, deshabilitar o habilitar servicios, así como verificar su estado y el de otras unidades del sistema.
    - Para iniciar un servicio: sudo systemctl start servicio.service
    - Para detener un servicio: sudo systemctl stop servicio.service
    - Para reiniciar un servicio: sudo systemctl restart servicio.service
    - Para comprobar el estado de un servicio: systemctl status servicio.service
    - Para habilitar un servicio al arranque: sudo systemctl enable servicio.service
4. En Systemd, un target (objetivo) es una unidad de systemd que agrupa varias unidades de servicio y otros targets para definir un estado final del sistema, como si fuera un grupo de servicios configurado para un propósito específico. Sirven un propósito similar a los runlevels de los sistemas SysVinit, gestionando el inicio o la detención de grupos de servicios relacionados como una sola unidad, facilitando así la organización de configuraciones del sistema.
    - default.target: El target predeterminado del sistema.
    - graphical.target: Inicia el sistema para un entorno de escritorio gráfico y multiusuario.
    - multi-user.target: Inicia el sistema en un estado multiusuario, sin entorno gráfico, pero con servicios de red activos.
    - reboot.target: Detiene el sistema y lo reinicia.
    - poweroff.target: Detiene el sistema por completo
5. Este comando muestra en la consola una jerarquía de procesos en forma de árbol. Todos los procesos salen del proceso padre systemd.

15.

1. La información principal sobre los usuarios se guarda en el archivo /etc/passwd, que contiene detalles como el nombre de usuario, ID de usuario, ID de grupo, directorio personal y shell predeterminada. La información de las contraseñas, cifrada, se almacena de forma segura en /etc/shadow. Los archivos de logs de actividad de los usuarios se encuentran en el directorio /var/log.

2. Las siglas UID (User Identifier) y GID (Group Identifier) hacen referencia a un número único que el sistema operativo Linux asigna a cada usuario y grupo, respectivamente, para identificar y controlar el acceso a los recursos del sistema. En un sistema Linux, los UIDs iguales sí pueden coexistir, pero no es recomendable, ya que un UID debe ser único para identificar a un usuario. Sin embargo, la existencia de UIDs idénticos es una mala práctica y una violación de los principios de seguridad del sistema operativo, que debe ser evitado en la medida de lo posible para garantizar la seguridad y el buen funcionamiento.
3. El usuario root es el superusuario en sistemas GNU/Linux, una cuenta con acceso y control total sobre el sistema, incluyendo la modificación de cualquier archivo y la administración de otros usuarios. No puede existir más de un usuario con el perfil de root, ya que su identificación se basa en un número de identificación único (UID). Específicamente, la UID de root es siempre 0.
4.
    i. sudo groupadd informatica
    ii. sudo useradd -m -g informatica isocso
    iii. sudo touch /home/isocso/test.txt
    iv. sudo chown isocso:informatica /home/isocso/test.txt
    v. sudo userdel -r isocso
    vi. grep isocso /etc/passwd
    vii. grep isocso /etc/group
    viii. sudo groupdel informatica
5.

Comando	**Funcionamiento**	**Parámetros**
useradd y adduser	useradd.- Crear un nuevo usuario o actualizar la información por defecto de un nuevo usuario. adduser.- Agregar o manipular usuarios.	useradd [options] LOGIN useradd -D useradd -D [options] adduser [--add-extra-groups] [--allow-all-names] [--allow-bad-names] [--comment comment] [--conf file] [--debug] [--disabled-login] [--disabled-password] [--firstgid id] [--firstuid id] [--gid id] [--home dir] [--ingroup group] [--lastgid id] [--lastuid id] [--no-create-home] [--shell shell] [--quiet] [--uid id] [--verbose] [--stdoutmsglevel prio] [--stderrmsglevel prio] [--logmsglevel prio] user adduser --system [--comment comment] [--conf file] [--debug] [--gid id] [--group] [--home dir] [--ingroup group] [--no-create-home] [--shell shell] [--uid id] [--quiet] [--verbose] [--stdoutmsglevel prio]

		[--stderrmsglevel prio] [--logmsglevel prio] user adduser --group [--conf file] [--debug] [--firstgid id] [--gid ID] [--lastgid id] [--quiet] [--verbose] [--stdoutmsglevel prio] [--stderrmsglevel prio] [--logmsglevel prio] group addgroup [--conf file] [--debug] [--firstgid id] [--gid ID] [--lastgid id] [--quiet] [--verbose] [--stdoutmsglevel prio] [--stderrmsglevel prio] [--logmsglevel prio] group addgroup --system [--gid id] [--conf file] [--quiet] [--verbose] [--stdoutmsglevel prio] [--stderrmsglevel prio] [--logmsglevel prio] group adduser [--conf file] [--debug] [--quiet] [--verbose] [--stdoutmsglevel prio] [--stderrmsglevel prio] [--logmsglevel prio] user group adduser --help adduser --version
usermod	Modificar la cuenta de un usuario.	usermod [options] LOGIN
userdel	Eliminar la cuenta y los archivos relacionados de un usuario.	userdel [options] LOGIN
su	Ejecutar un comando con ID de usuario y grupo sustitutos.	su [options] [-] [user [argument...]]
groupadd	Crear un nuevo grupo.	groupadd [OPTIONS] NEWGROUP
who	Muestra quién está conectado al sistema.	who [OPCIÓN]... [FICHERO | ARG1 ARG2]
groupdel	Borrar un grupo.	groupdel [options] GROUP
passwd	Cambiar la contraseña de un usuario.	passwd [options] [LOGIN]

16.

1. Los permisos sobre archivos se definen a través de tres tipos de permisos para tres tipos de usuarios:

i. **r (lectura)**: Permite leer el contenido de un archivo. En el caso de un directorio, permite listar los archivos que contiene.

ii. **w (escritura)**: Permite modificar o borrar el contenido de un archivo. En un directorio, permite crear, renombrar o borrar archivos dentro de él.

iii. **x (ejecución)**: Permite ejecutar un archivo como un programa o *script*. En un directorio, permite entrar en él (`cd`).

Estos tres permisos se aplican a tres categorías de usuarios:

- **Propietario (u)**: El dueño del archivo, que generalmente es el usuario que lo creó.
- **Grupo (g)**: El grupo al que pertenece el archivo. Todos los miembros de este grupo tienen los permisos asignados a esta categoría.
- **Otros (o)**: Cualquier usuario que no sea el propietario ni pertenezca al grupo.

Los permisos se visualizan a través del comando `ls -l` y se representan en una cadena de 10 caracteres.

- El **primer carácter** indica el tipo de archivo (ej: `-` para archivo regular, `d` para directorio, `l` para enlace simbólico).
- Los siguientes **nueve caracteres** se dividen en tres grupos de tres, cada uno para el propietario, el grupo y otros, en ese orden.

Por ejemplo, la representación de permisos `-rwxr-xr--` significa:

- `-`: Es un archivo regular.
- `rwx`: El **propietario** tiene permisos de lectura, escritura y ejecución.
- `r-x`: El **grupo** tiene permisos de lectura y ejecución, pero no de escritura.
- `r--`: **Otros** solo tienen permisos de lectura.

2.

i. chmod: modifica los permisos de acceso de cada archivo conforme a modo, que puede ser una representación simbólica de dichas modificaciones o un número octal que representa el patrón de bits para el nuevo modo.

ii. chown: modifica la propiedad del usuario y/o grupo de cada archivo dado. Si solo se proporciona un propietario (un nombre de usuario o ID de usuario numérico), ese usuario se convierte en el propietario de cada archivo dado y el grupo al que pertenecen los archivos no se modifica. Si el propietario va seguido de dos puntos y un nombre de grupo (o ID de grupo numérico), sin espacios entre ellos, también se cambiará el grupo al que pertenecen. Si se ponen dos puntos pero ningún nombre de grupo a continuación, ese usuario se convierte en el propietario de los archivos y el grupo de archivos se cambia al grupo principal al que pertenece ese usuario. Si se indican los dos puntos y el grupo, pero se omite el propietario, solo se cambiará el grupo; en este caso, chown realiza la misma función que chgrp. Si

solo se dan dos puntos, o si todo el operando está vacío, no se cambia ni el propietario ni el grupo.

iii. Cambia el grupo de cada ARCHIVO a GRUPO. Si añade --reference, cambiará el grupo de cada ARCHIVO al del R-ARCHIVO.

3. Un modo numérico consta de entre uno y cuatro dígitos **octales** (0-7), que se obtienen sumando los bits con los valores 4, 2 y 1. Los dígitos que faltan se consideran ceros y se colocan al incio. El primer dígito selecciona el ID de usuario configurado (4), el ID de grupo configurado (2), por último, los atributos de eliminación restringida o sticky bit (1). El segundo dígito selecciona los permisos para el propietario del archivo: leer (4), escribir (2) y ejecutar (1); el tercero selecciona permisos para otros usuarios en el grupo del archivo y el cuarto para otros usuarios que no están en el grupo del archivo, ambos con los mismos rangos de valores.

chmod 754 archivo: Asigna rwx al propietario, r-x al grupo y r-- a otros.

4. Existe la posibilidad de que un usuario pueda acceder a un archivo para el cual no tiene permisos, si otro usuario con los permisos adecuados le otorga acceso de forma temporal o permanente. También si este tiene privilegios de superusuario.

5. “Full Path Name” hace referencia a la dirección exacta de un archivo o directorio desde que comienza el directorio raíz (/).

“Relative Path Name” es la dirección de un archivo o directorio que se especifica en relación a la ubicación actual del usuario.

Ejemplo de Ruta Absoluta (comienza con /):

Si se desea acceder a un archivo llamado reporte.pdf dentro del directorio documentos, su ruta absoluta es:

/home/usuario/documentos/reporte.pdf

Si se quiere acceder al directorio de configuración del sistema etc, su ruta absoluta es:

/etc

Ejemplo de Ruta Relativa (no comienza con /):

Para acceder al mismo archivo reporte.pdf desde /home/usuario/documentos, la ruta relativa es simplemente:

reporte.pdf

Para acceder a un archivo llamado apuntes.txt que está en /home/usuario/, la ruta relativa desde tu ubicación actual es:

../apuntes.txt

El .. significa "subir un nivel de directorio".

6. Con “pwd” se puede saber en qué directorio se está actualmente.
Con “cd ~” se puede acceder al directorio personal.
“cd .” mantiene al usuario en el mismo directorio, mientras que “cd ..” lo hace subir de nivel, es decir, al directorio “anterior”.

7.

Comando	**Funcionalidad**	**Parámetros**
umount	Desmontar filesystems.	umount -a [-dflnrv] [-t fstype] [-O option...] umount [-dflnrv] {directory|device} umount -h|-V
du	Estima el uso de espacio de ficheros.	du [OPCIÓN]... [FICHERO]... du [OPCIÓN]... --files0-from=F
df	Informa de la utilización del espacio de disco en sistemas de archivo.	df [OPCIÓN]... [FICHERO]...
mount	Montar un filesystem.	mount [-h|-V] mount [-l] [-t fstype] mount -a [-fFnrsvw] [-t fstype] [-O optlist] mount [-fnrsvw] [-o options] device|mountpoint mount [-fnrsvw] [-t fstype] [-o options] device mountpoint mount --bind|--rbind|--move olddir newdir mount --make-[shared|slave|private|unbindable|rshared|rslave|rprivate|runbindable] mountpoint
mkfs	Construye un sistema de ficheros de Linux.	mkfs [options] [-t type] [fs-options] device [size]
fdisk	Manipula la tabla de particiones del disco.	fdisk [options] device fdisk -l [device...]

write	Enviar un mensaje de texto a otro usuario que está actualmente conectado al mismo sistema.	write usuario [terminal]
losetup	Configurar y controlar dispositivos de bucle.	Get info: losetup [loopdev] losetup -l [-a] losetup -j file [-o offset] Detach a loop device: losetup -d loopdev ... Detach all associated loop devices: losetup -D Set up a loop device: losetup [-o offset] [--sizelimit size] [--sector-size size] [--loop-ref name] [-Pr] [--show] -f|loopdev file Resize a loop device: losetup -c loopdev
stat	Muestra el estado de un archivo o sistema de archivos.	stat [OPCIÓN]... ARCHIVO...

17.

1. Un proceso que se ejecuta en **foreground** es aquel que está en primer plano y tiene el control de la terminal. Mientras un proceso está en foreground, no puedes ingresar nuevos comandos hasta que el proceso termine o lo detengas. Por el contrario, un proceso en **background** es aquel que se ejecuta en segundo plano, liberando la terminal para que puedas seguir ingresando comandos.
2. **Para ejecutar un proceso en background**, simplemente agrega un ampersand (&) al final del comando. Por ejemplo:

    sleep 30 &

Este comando hará que el sistema espere durante 30 segundos en segundo plano, mientras puedes seguir usando la terminal.

Para pasar un proceso de background a foreground, se usa el comando `fg` (foreground). Si se tienen varios procesos en background, se puede especificar cuál se quiere traer al frente. Primero, se usa el comando `jobs` para ver los procesos en segundo plano. Luego, se usa `fg %ID_DEL_PROCESO` para traerlo a primer plano. Por ejemplo:

```
jobs
# [1]+  Running                 sleep 30 &
fg %1
```

Para pasar un proceso de foreground a background, primero se debe detenerlo temporalmente presionando la combinación de teclas `Ctrl + Z`. Luego, se usa el comando `bg` (background) para enviarlo a segundo plano:

```
sleep 30
# Presiona Ctrl + Z
# [1]+  Stopped                 sleep 30
bg
```

3. La **pipe** (`|`) es un operador que se utiliza para conectar la salida de un comando con la entrada de otro. Esto permite encadenar comandos para crear flujos de trabajo más complejos y potentes. La finalidad de una pipe es usar el resultado de una operación como el punto de partida para la siguiente.

   Ejemplos de utilización:

- **Contar archivos en un directorio**: `ls -l` lista el contenido de un directorio. Al usar `| wc -l`, la salida de `ls -l` se redirige al comando `wc` (word count), que cuenta el número de líneas, dándote el total de archivos.

  ```
  ls -l | wc -l
  ```

- **Buscar texto en la salida de un comando**: `ps -ef` muestra todos los procesos del sistema. Al usar `| grep ssh`, solo verás las líneas que contengan la cadena "ssh".

  ```
  ps -ef | grep ssh
  ```

4. La redirección es el proceso de cambiar la entrada o salida estándar de un comando. En lugar de que la salida se muestre en la terminal, puedes enviarla a un archivo, o puedes hacer que un comando reciba su entrada desde un archivo en lugar de desde el teclado.

   Existen **tres** tipos principales de redirección:

   - **Redirección de salida (> y >>)**:

     >: Redirige la salida estándar a un archivo, **sobrescribiendo** su contenido si ya existe.

     ```
     echo "Hola mundo" > archivo.txt
     ```

>>: Redirige la salida estándar a un archivo, **añadiendo** el contenido al final del archivo.

```
echo "Adiós mundo" >> archivo.txt
```

- **Redirección de entrada (<)**:

  Redirige la entrada estándar de un comando para que la tome desde un archivo en lugar de desde el teclado.

  ```
  wc -l < archivo.txt
  ```

  Esto le dice a `wc` que cuente las líneas dentro de `archivo.txt`.

- **Redirección de error (2>)**:

  Redirige la salida de errores estándar a un archivo, lo que es muy útil para depurar *scripts*. El número 2 representa el *file descriptor* para la salida de error.

  ```
  ls -l archivo_que_no_existe 2> error.log
  ```

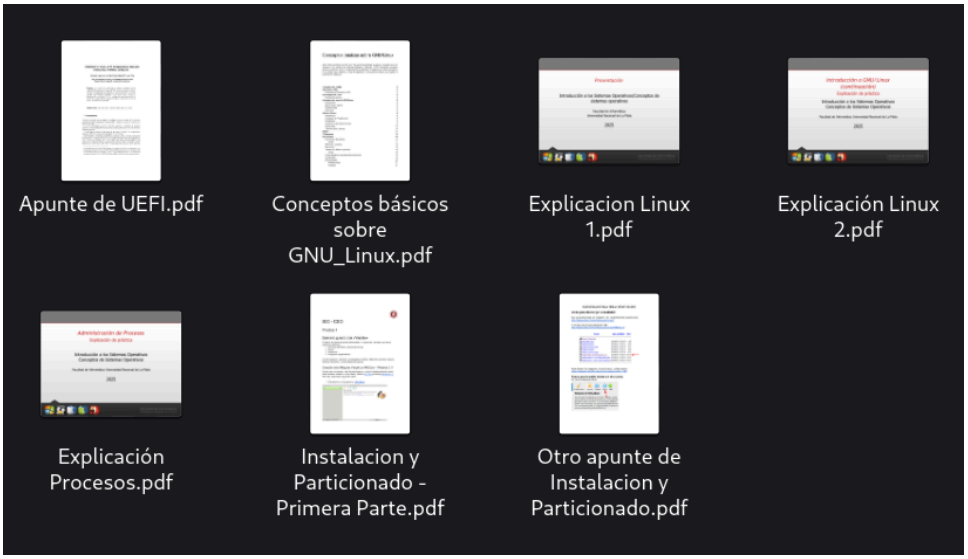
  El error "no existe el archivo" se escribirá en `error.log` en lugar de en la terminal.

18.

1. Empaquetar archivos en GNU/Linux se refiere al proceso de agrupar múltiples archivos y directorios en un solo archivo. A diferencia de la compresión, que reduce el tamaño de los datos, el empaquetado simplemente crea un "contenedor" o "paquete" que facilita la manipulación, el almacenamiento y la transferencia de un conjunto de archivos.

   El comando más utilizado para empaquetar archivos es tar (de Tape Archive). Por sí solo, tar no comprime los archivos, solo los agrupa.

2. Tomaré estos archivos:



   En total, suman 4,9 MB. Al empaquetarlos, usando tar y gzip, me queda un paquete de 4,4 MB.

3. tar -czvf mi_paquete.tar.gz "Apunte de UEFI.pdf" "Conceptos básicos sobre GNU_Linux.pdf" "Explicacion Linux 1.pdf" "Explicación Linux 2.pdf"

"Explicación Procesos.pdf" "Instalacion y Particionado - Primera Parte.pdf" "Otro apunte de Instalacion y Particionado.pdf"

4. Sí, se puede.
5.

Comando	**Funcionalidad**
tar	Es un programa de archivado diseñado para almacenamiento de múltiples archivos en un solo archivo (un archive) así como su manipulación. El archivo puede ser un archivo normal o un dispositivo (por ejemplo, una unidad de cinta, de ahí el nombre del programa, que viene de 'tape archiver', que puede ubicarse en el propio equipo o en un equipo remoto.
grep	Busca PATRONES en cada ARCHIVO. PATRONES consistirá en uno o más patrones separados entre sí por un salto de línea; grep mostrará cada línea donde encuentre una concordancia con dicho patrón. En general, deben entrecomillarse los patrones si se ejecuta grep dentro de otra orden de la shell.
gzip	Reduce el tamaño de los archivos nombrados mediante la codificación Lempel-Ziv (LZ77). Siempre que es posible, cada archivo se reemplaza por uno con la extensión .gz, manteniendo los mismos modos de propiedad, acceso y tiempos de modificación.
zgrep	Se emplea para invocar grep sobre ficheros comprimidos o "gzipeados". Todas las opciones especificadas se pasan directamente a grep. Si no se especifica ningún fichero, entonces se descomprime (si es necesario) la entrada estándar y se envía a grep. En otro caso, se descomprimen los archivos dados, si es necesario, y se envían a grep.
wc	Muestra el total de líneas, palabras y bytes de cada ARCHIVO y una totalización si se especifica más de un ARCHIVO. Se considera una

	palabra a una cadena de más de cero caracteres delimitada por espacios o por el inicio y final de una entrada.

19.

Comando	**Resultado**
l s −l > prueba	Se creó un archivo llamado “prueba” y se escribió en el mismo archivo: “total 0 -rw-rw-r-- 1 fededobal-noroot fededobal-noroot 0 ago 28 10:29 prueba”
ps > PRUEBA	Se creó el archivo “PRUEBA” y se escribió en el mismo archivo: “ PID TTY TIME CMD 6229 pts/0 00:00:00 bash 6581 pts/0 00:00:00 bash 7312 pts/0 00:00:00 ps “
chmod 710 prueba	El archivo “prueba” tiene permisos rwx--x- - -. Es decir, RWX para el propietario, X para el grupo, y nada para otros.
chown root:root PRUEBA	chown: cambiando el propietario de 'PRUEBA': Operación no permitida.
chmod 777 PRUEBA	El archivo “PRUEBA” ahora tiene permisos rwx- -rwx- -rwx. Es decir, todos los permisos para todos.
chmod 700 /etc/passwd	chmod: cambiando los permisos de '/etc/passwd': Operación no permitida.
passwd root	passwd: no debe ver o cambiar la información de la contraseña para root.
rm PRUEBA	Se borra el archivo “PRUEBA”.
man /etc/shadow	man: can't open /etc/shadow: Permiso denegado.
find / −name ∗ .conf	Busca y lista todos los archivos con extensión .conf desde el directorio raíz. Tira muchos “Permiso denegado” (casi me revienta la máquina).
usermod root −d /home/ newroot −L	Permiso denegado. Por defecto, un usuario no tiene permisos 700 cómo para acceder al directorio de root.

cd /root	Permiso denegado por lo mismo.
rm *	Se borran todos los archivos posibles (prueba era el único que quedaba).
cd /etc	Se accede al directorio /etc.
cp * /home -R	Falla.
shutdown	Falla. Orden no encontrada.

20.

1. sudo kill 23
2.
    i. sudo killall init -> No encuentra el proceso “init”
    ii. sudo killall systemd -> Cierra sesión
3. find /home -name “*.conf”
4.
    i. cd /home/fededobal
    ii. mkdir procesos
    iii. cd procesos
    iv. ps > procesos.txt
5.
    i. touch xxxx
    ii. chmod 751 xxxx
6.
    i. touch yyyy
    ii. chmod 650 yyyy
7.
    i. cd /tmp
    ii. rm *
8.
    i. cd /opt
    ii. sudo mkdir isodata
    iii. sudo chown isocso isodata
9.
    i. cd ~
    ii. touch donde (si no existe)
    iii. pwd >> donde

21.

1. su root
2. sudo adduser fdobal
3. Se crearon, en /home/fdobal/:
    i. .bash_logout
    ii. .bashrc
    iii. .face
    iv. .face.icon
    v. .profile
4.

   i. cd /tmp
   ii. mkdir miCursada
5. cp -r /var/log/* /tmp
6.
   i. cd /tmp
   ii. chown fdobal miCursada
   iii. chgrp users miCursada
7. chmod -R 777 miCursada
8. su fdobal
9. hostname
10. ps -ef
11. who
12.
   i. su -
   ii. write fdobal “se te va a apagar el sistema”
13. shutdown

22.

1. mkdir -p "25983"
   cd 25983
2. vim LEAME
   i
   escribo mis datos
   Escape
   :wq
3. chmod 017 LEAME
4. cd /etc
   ls
   ls > ~/leame
   Esto se puede hacer porque la shell es case-sensitive, es decir, diferencia entre “LEAME” y “leame”.
5. sudo find / -name "leame"
   sudo find / -name "*.txt" *Busca todos los archivos con extensión .txt.*
6. sudo find / -name "*.so" > ~/25983/ejercicioF

23.

1. crea el directorio “iso” en donde se está ubicado.
2. mueve al subdirectorio iso y luego crea o sobrescribe un archivo llamado f0 con una lista de los procesos que se están ejecutando.
3. exporta al archivo “f1” el listado de los ficheros y archivos presentes.
4. se mueve al dir. raíz.
5. imprime la ruta del directorio personal del usuario actual.
6. lista los archivos en el directorio actual y guardaría la salida en un archivo llamado ls dentro de ~/iso.
7. mueve el directorio actual al directorio personal del usuario y luego crea un nuevo directorio llamado f2.
8. muestra información detallada sobre el directorio f2 sin listar su contenido.
9. cambia los permisos del directorio f2 a 3 (escritura y ejecución) para el propietario, 4 (lectura) para el grupo y 1 (ejecución) para otros.
10. crea un archivo vacío llamado dir en el directorio actual.

11. mueve el directorio actual a f2.
12. mueve el directorio actual al subdirectorio iso dentro del directorio personal del usuario.
13. muestra la ruta del directorio actual y la guarda en un nuevo archivo llamado f3, sobrescribiendo cualquier contenido anterior.
14. muestra los procesos, filtra los que contienen "ps", cuenta el número de líneas resultantes y añade ese número al final del archivo f3 que se encuentra en el directorio f2.
15. cambia los permisos del directorio f2 a 700 (rwx para el propietario, y sin permisos para los demás) y luego mueve el directorio actual un nivel arriba.
16. busca un archivo llamado etc/passwd dentro del directorio actual, lo cual probablemente no encuentre nada.
17. busca un archivo llamado etc/passwd en todo el sistema de archivos, lo cual debería encontrar la ruta /etc/passwd.
18. crea un nuevo directorio llamado ejercicio5 en la ubicación actual.