1. Características de GNU/Linux

Ejercicio 1.a

Mencione y explique las características más relevantes de GNU/Linux

- **Es multiusuario:** distintos usuarios pueden usar un mismo sistema a la vez.
- **Es multitarea y multiprocesador.**
- **Es altamente portable:** el mismo código fuente puede compilarse para distintas plataformas.
- **Posee diversos comandos de los cuales algunos son programables.**
- **Permite el manejo de usuarios y permisos**
- **Todo es un archivo (hasta los dispositivos y directorios)**
- **Cada directorio puede estar en una partición diferente.**
- **Es case-sensitive**
- **Es código abierto**

Ejercicio 1.b

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

Windows:

- **Es multiusuario** en sus versiones empresariales (Server) las versiones domésticas tienen capacidades multiusuario muy limitadas.
- **Es multitarea y multiprocesador.**
- **No es muy portable:** por lo menos es menos portable que GNU/Linux.
- **Posee menor cantidad de comandos** que Linux.
- **Permite el manejo de usuarios y permisos:** pero su uso es más complejo y menos transparente.
- **Casi todo es un objeto:** muchos de los recursos se representan como objetos, pero esta filosofía no es tan marcada como la de los archivos en GNU/Linux.
- **Tiene unidades de disco:** a diferencia de GNU/Linux, que parte de un *“árbol de directorios”,* donde todos los archivos y directorios parten del directorio raíz "/";, Windows tiene varias unidades de disco, generalmente etiquetadas como C:, D:, E:, etc. y cada una de estas unidades de disco puede considerarse como su propio “punto de inicio” en la estructura de directorios. Sin embargo, en Windows también es posible montar unidades en un directorio, aunque no es tan común hacerlo.
- **No es case-sensitive**, y existen algunas limitaciones para los nombres de los archivos y directorios (caracteres y nombres inválidos).
- **Es software propietario:** el acceso al código fuente está restringido y controlado por Microsoft.

Ejercicio 1.c

¿Qué es GNU?

> GNU es un sistema operativo de software libre, es decir, respeta la libertad de los usuarios. El sistema operativo GNU consiste en paquetes de GNU (programas publicados específicamente por el proyecto GNU) además de software libre publicado por terceras partes. El desarrollo de GNU ha permitido que se pueda utilizar un ordenador sin software que atropelle nuestra libertad.
>
> GNU es un sistema operativo de tipo Unix, lo cual significa que se trata de una colección de muchos programas: aplicaciones, bibliotecas, herramientas de desarrollo y hasta juegos. El desarrollo de GNU, iniciado en enero de 1984, se conoce como Proyecto GNU. Muchos de los programas de GNU se publican bajo el auspicio del Proyecto GNU y los llamamos paquetes de GNU.
>
> El nombre «GNU» es un acrónimo recursivo de «GNU No es Unix». En un sistema de tipo Unix, el programa que asigna los recursos de la máquina y se comunica con el hardware se denomina «kernel» (o núcleo). GNU se usa generalmente con un kernel llamado «Linux». Esta combinación es el sistema operativo GNU/Linux. Millones de personas usan GNU/Linux, aunque muchos lo llaman erróneamente «Linux».
>
> Sitio oficial de GNU

Ejercicio 1.d

Indique una breve historia sobre la evolución del proyecto GNU

> El desarrollo de GNU ha sido liderado desde su inicio en 1983 por Richard Stallman, con el fin de crear un sistema operativo libre tipo Unix (Unix-like), el sistema GNU. Para asegurar que fuera libre, se creó un marco regulatorio conocido como GPL (GNU General Public License)
>
> En 1990, GNU ya contaba con un editor de textos (Emacs), un compilador (GCC) y una gran cantidad de bibliotecas que componen un Unix típico; pero faltaba el componente principal, el núcleo (kernel).
>
> Si bien ya se venía trabajando en un núcleo conocido como *TRIX*, en 1988 se decide abandonarlo debido a su complejidad y se decide adoptar como base el núcleo *MACH* para crear *GNU Hurd*, el cual tampoco prosperó.
>
> Linus Torvalds venía trabajando desde 1991 en un Kernel denominado Linux, el cual se distribuía bajo licencia GPL. En el año 1992, Torvalds y Stallman deciden fisionar ambos proyectos y es allí donde nace GNU/Linux.
>
> Más cosas: Cuál es la diferencia entre Linux y GNU/Linux.

Ejercicio 1.e

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

La **multitarea** describe la capacidad de ejecutar muchos programas al mismo tiempo sin detener la ejecución de cada aplicación.

Cada núcleo del procesador solo es capaz de hacer una tarea a la vez, pero la realiza en tiempos tan cortos (< 20 ms) que no lo notamos. Un sistema operativo multitarea divide el tiempo de procesador disponible entre los procesos o subprocesos que lo necesitan. El sistema está diseñado para la multitarea preferente: asigna un segmento de tiempo de procesador a cada subproceso que ejecuta. El subproceso que se está ejecutando actualmente se suspende cuando transcurre su segmento de tiempo, lo que permite que se ejecute otro subproceso. Cuando el sistema cambia de un subproceso a otro, guarda el contexto del subproceso adelantado y restaura el contexto guardado del siguiente subproceso de la cola.

Linux se basa en la multitarea prioritaria, donde cada programa tiene garantizada la oportunidad de ejecutarse, y se ejecuta hasta que el sistema operativo da prioridad a otro programa para ejecutarse.

Características del S.O. Linux
Multitarea - Microsoft learn

Ejercicio 1.f

¿Qué es POSIX?

Hace más de 40 años, los programadores tenían que reescribir su código completamente si querían adaptarlo de un sistema a otro. Para resolver este problema de compatibilidad y que fuera posible escribir programas que pudieran ejecutarse en cualquier sistema UNIX, en la década de 1980, el IEEE desarrolló el estándar **IEEE 1003.1** para UNIX, conocido como **POSIX (Portable Operating System Interface)**.

POSIX define una interfaz mínima de **llamadas al sistema** que los sistemas UNIX deben seguir. En realidad, define un conjunto de procedimientos de biblioteca que deben proporcionar todos los sistemas UNIX que sigan el estándar. La mayoría de estos procedimientos invocan una llamada al sistema, aunque algunos se pueden implementar fuera del kernel.

La idea detrás de POSIX es que un desarrollador de software que escriba un programa utilizando únicamente los procedimientos definidos por él pueda estar seguro de que se ejecutará en todos los sistemas UNIX que también sigan el estándar.

POSIX no se creó para controlar como se construyen los sistemas operativos: cualquier empresa es libre de diseñar su variante de UNIX como quiera. POSIX solo se preocupa de cómo una aplicación interactúa con el sistema operativo a través de las llamadas al sistema. Así, cuando un programa se adecua al estándar POSIX, es más fácil adaptarlo a otro sistema que también respete este estándar, teniendo que reescribir muy poco código, si es que siquiera se necesita reescribir alguno.

Tanenbaum, A. S. (2007). Introducción. Historia de los sistemas operativos. En *Sistemas operativos modernos* (3ra ed., capítulo 10, p. 14). Prentice Hall.

Tanenbaum, A. S. (2007). Caso de estudio 1: Linux. Unix estándar. En *Sistemas operativos modernos* (3ra ed., capítulo 10, pp. 724-725). Prentice Hall.

What is POSIX? - It's Foss

2. Distribuciones de GNU/Linux

Ejercicio 2.a

¿Qué es una distribución GNU/Linux? Nombre al menos 4 distribuciones de GNU/Linux y cite diferencias básicas entre ellas.

> Una distribución de GNU/Linux es un sistema operativo compilado a partir de componentes desarrollados por varios proyectos de código abierto que se combinan para formar un único sistema operativo que se puede instalar y arrancar. Cada distribución incluye el kernel de Linux, módulos de software del proyecto GNU, un gestor de paquetes, un programa de instalación, utilidades, aplicaciones y, en la mayoría de los casos, un entorno de escritorio junto con un gestor de ventanas para proporcionar un sistema operativo completo y funcional.}
>
> Cada distribución puede estar diseñada con objetivos y enfoques diferentes, lo que lleva a variaciones en la experiencia de usuario y en las características disponibles. Al tratarse de software de código abierto, cualquiera puede crear su propia distribución de Linux compilándola a partir del código fuente o modificando una distribución ya existente. La elección de una distribución depende de las necesidades y preferencias individuales, así como del nivel de experiencia técnica del usuario. Cada distribución tiene sus propias características, ventajas y desventajas, por lo que es importante seleccionar la que mejor se adapte a nuestras necesidades específicas.
>
> En la actualidad se mantienen activamente más de 300 distribuciones de GNU/Linux. Existen distribuciones de GNU/Linux para computadoras de escritorio, servidores (sin interfaz gráfica), supercomputadoras, dispositivos móviles y para usos especiales, como sistemas embebidos.
>
> What is a Linux distribution? - SUSE
> Definition of Linux distribution - PCMag

Una visión general de las principales distribuciones de Linux según DistroWatch acá.

Ejercicio 2.b

¿En qué se diferencia una distribución de otra?

- **Núcleo y versión del kernel:** si bien todas las distribuciones de GNU/Linux usan el kernel de Linux, pueden variar en la versión que utilizan.
- **Sistema de gestión de paquetes:** las distribuciones usan diferentes sistemas de gestión de paquetes para instalar, actualizar y administrar software. Debian, Ubuntu y derivados usan APT; Fedora y RHEL usan DNF; Arch Linux usa pacman; etc.

- **Conjunto de herramientas y entornos de escritorio:** las distribuciones pueden incluir diferentes conjuntos de herramientas y entornos de escritorio por defecto. Por ejemplo, Fedora utiliza GNOME, mientras que Linux Mint utiliza Cinnamon.
- **Filosofía y objetivos:** las distribuciones de GNU/Linux suelen crearse con objetivos específicos en mente. Algunas están diseñadas para ser extremadamente ligeras y eficientes (como Puppy Linux), mientras que otras se enfocan en la seguridad (como Kali Linux) o en la estabilidad a largo plazo (como Debian).

Ejercicio 2.c

¿Qué es Debian? Acceda al sitio e indique cuáles son los objetivos del proyecto y una breve cronología del mismo.

> El Proyecto Debian es una asociación de individuos que se han unido para crear un sistema operativo libre llamado Debian. Los sistemas Debian actualmente usan el kernel Linux o el kernel FreeBSD. Sin embargo, se está trabajando para proveer Debian para otros kernels, principalmente a Hurd. Además, gran parte de las herramientas básicas que completan al sistema operativo vienen del proyecto GNU.
>
> Debian viene con más de 59000 paquetes (software precompilado que se empaqueta en un formato fácil de instalar) (deb), un gestor de paquetes (APT) y otras utilidades que hacen posible administrar miles de paquetes en miles de computadoras de forma tan sencilla como instalar una sola aplicación.
>
> About Debian - Debian

> El proyecto Debian fue fundado por Ian Murdock el 16 de agosto de 1993. La primera versión de Debian (0.01) se publicó el 15 de septiembre de 1993 y su primera versión estable (1.1) se publicó el 17 de junio de 1996. Esto convierte a Debian es uno de los sistemas operativos más antiguos basados en el núcleo de Linux, y actualmente es la segunda distribución de Linux más antigua aún en desarrollo activo, solo por detrás de Slackware. Debian es también la base de muchas distribuciones, entre las que se destaca Ubuntu; del cual, a su vez, se desprenden otras distribuciones como Kubuntu, Xubuntu, Lubuntu, Linux Mint, etc.
>
> Debian - Wikipedia

Historia completa: A brief history of Debian - Debian.

3. Estructura de GNU/Linux

Ejercicio 3.a

Nombre cuáles son los tres componentes fundamentales de GNU/Linux.

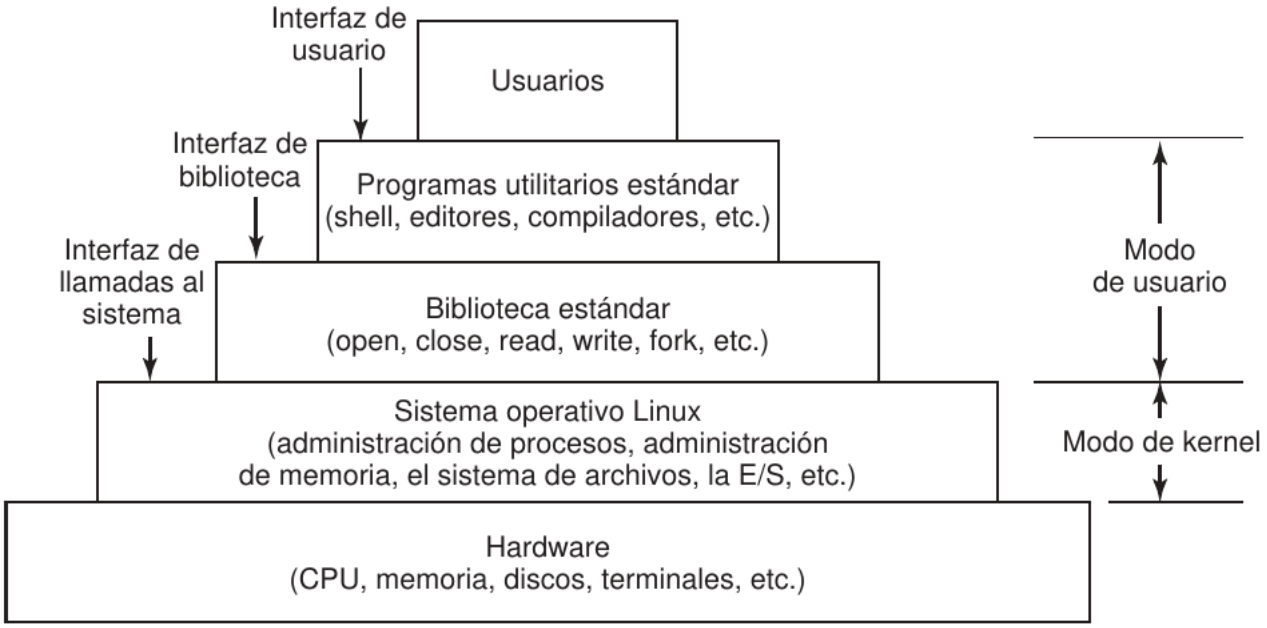
En conjunto, el kernel de Linux, las bibliotecas GNU y las utilidades GNU forman la base del sistema operativo GNU/Linux.

- **Kernel:** es el encargado de que el software y el hardware puedan trabajar juntos, y para hacerlo administra la memoria, la CPU y la E/S. En los sistemas GNU/Linux se utiliza el kernel Linux, que es un kernel monolítico híbrido (lo que lo hace híbrido es la capacidad de cargar y descargar funcionalidades a través de módulos).
- **Sistema de Bibliotecas GNU (GNU Libraries):** a menudo son llamadas GNU C Library (glibc) y proporcionan funciones esenciales y rutinas que los programas y aplicaciones utilizan para interactuar con el kernel y realizar diversas tareas. Las bibliotecas GNU incluyen funciones para la gestión de archivos, la comunicación en red, la administración de memoria y muchas otras operaciones esenciales.
- **Utilidades GNU (GNU Utilities):** son un conjunto de programas y comandos que proporcionan herramientas esenciales para la administración y el funcionamiento del sistema. Estas utilidades incluyen comandos de línea de comandos como ls, cp, mv, rm, mkdir, y muchos otros. Estas herramientas permiten a los usuarios y administradores realizar tareas como navegar por el sistema de archivos, copiar y mover archivos, administrar procesos y configurar el sistema.
- **Intérprete de comandos:** también conocido como interfaz de línea de comandos (command-line interface, CLI). Es el modo de comunicación entre el usuario y el SO, cada usuario puede tener una interfaz, o shell, desde donde puede ejecutar programas a partir del ingreso de comandos. Además, existen distintos intérpretes de comandos para Linux: Bash (Bourne-Again Shell), Zsh (Z Shell), Fish (Friendly Interactive Shell), Ksh (Korn Shell), etc.
- **Sistema de archivos:** organiza la forma en que se almacenan los archivos en dispositivos de almacenamiento. GNU/Linux soporta una amplia gama de formatos (ext4, btrfs, fat, ntfs, ext3, ext2, etc.). Además, los archivos en Linux se organizan según FHS (Filesystem Hierarchy Standard).

Ejercicio 3.b

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

Un sistema Linux se puede considerar un tipo de pirámide donde en la parte inferior está el hardware, que consiste en CPU, memoria, discos, un monitor, teclado y otros dispositivos. En el hardware básico se ejecuta el sistema operativo. Su función es controlar el hardware y proveer una interfaz de llamadas al sistema para todos los programas. Estas llamadas al sistema permiten a los programas de usuario crear y administrar procesos, archivos y otros recursos.



Tanenbaum, A. S. (2007). Caso de estudio 1: Linux. En *Sistemas operativos modernos* (3ra ed., capítulo 10, p. 730). Prentice Hall.

4. Kernel

Ejercicio 4.a

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

> En 1991 Linus Torvalds inicia la programación de un Kernel basado en Minix y el 5 de octubre de ese mismo año anuncia la primera versión “oficial” de Linux (la versión 0.02). En 1992 combina su desarrollo con GNU formando GNU/Linux y el 14 de marzo de 1994 se lanza la versión 1.0 del kernel. Desde entonces, el desarrollo del kernel de Linux continúa junto a miles de programadores al rededor del mundo.

Ejercicio 4.b

¿Cuáles son sus funciones principales?

> El kernel es el núcleo del sistema operativo (en sentido estricto, es el sistema operativo) y es el encargado de ejecutar programas y gestionar los dispositivos de hardware. Se encarga de que el software y el hardware puedan trabajar juntos, y para hacerlo administra la memoria, la CPU y la E/S. En los sistemas GNU/Linux se utiliza el kernel Linux.

Ejercicio 4.c

¿Cuál es la versión actual? ¿Cómo se definía el esquema de versionado en versiones anteriores a la 2.4? ¿Qué cambió en el versionado se impuso a partir de la versión 2.6?

> Las versiones del Kernel Linux pueden tener 4 números que indican la versión: **`a.b.c.d`**
>
> a. indica la **versión**. Este número es el que menos cambia, ya que solo se suele dar el salto cuando hay cambios extremadamente grandes en el sistema. En toda su historia, solo ha cambiado 5 veces, en 2004, para la versión 1.0, en 2006, para la versión 2.0, en 2011, para la versión 3.0, en 2015, para la versión 4.0, en 2019 para la versión 5.0 y a finales de 2022 para dar lugar a la versión actual, la 6.0 (actualmente la última versión es la 6.5.2).
> b. indica la **subversión**. Cuando se lanzan nuevas versiones, pero realmente son actualizaciones menores (nuevos drivers, optimizaciones, correcciones, etc), entonces en lugar de cambiar la versión, se cambia el número de la subversión.
> c. indica el nivel de **revisión**. Se suele cambiar este número, por ejemplo, cuando se introducen cambios menores, como parches de seguridad, revisiones de errores, etc.
> d. es el último **subnivel** de la versión. Apenas se utiliza, pero está reservado para que, si se lanza una versión con un fallo muy grave, se lance la nueva versión con este subnivel incluyendo exclusivamente la corrección de dicho fallo grave.
>
> Kernel Linux, descubre cómo es el corazón de este sistema operativo - SoftZone

Ejercicio 4.d

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

Si, es posible tener más de un kernel instalado en la misma máquina, pero solo puede haber uno activo a la vez. Como un kernel nuevo solo puede evaluarse mediante el inicio en ese kernel, el paquete está específicamente diseñado para que puedan instalarse múltiples versiones simultáneamente. Así, si el kernel nuevo no arranca, el kernel anterior sigue estando disponible.

Cuando nuestra distribución de GNU/Linux baja una actualización del Kernel, esta no baja solo las partes que han cambiado, sino que baja de nuevo el núcleo completo en el sistema, y lo instala completo. Además, para evitar problemas, deja todas las versiones antiguas del mismo guardadas en el disco duro, de manera que, si la nueva versión no funciona correctamente, podamos arrancar una versión anterior del núcleo desde el menú de arranque de GRUB.

Red Hat (2017). Paquetes de software y RPM. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 13, pp. 347). Red Hat.
Kernel Linux, descubre cómo es el corazón de este sistema operativo - SoftZone

Ejercicio 4.e

¿Dónde se encuentra ubicado dentro del File System?

El kernel de Linux se encuentra generalmente en el directorio /boot/ y se llama “vmlinuz” o “bzImage” seguido de la versión del kernel. Además, muchas distribuciones guardan los archivos fuente de las distintas versiones de kernel instaladas en el directorio /usr/src/linux.

Ejercicio 4.f

¿El kernel de GNU/Linux es monolítico? Justifique su respuesta.

El kernel de Linux es **monolítico híbrido.** Aunque Linux no utiliza una técnica de micronúcleo, logra muchas de las ventajas potenciales de esta técnica por medio de su arquitectura modular particular. Linux está estructurado como una colección de módulos, algunos de los cuales pueden cargarse y descargarse automáticamente bajo demanda. Estos bloques relativamente independientes se denominan **módulos cargables**. Esencialmente, un módulo es un fichero cuyo código puede enlazarse y desenlazarse con el núcleo en tiempo real. Normalmente, un módulo implementa algunas funciones específicas, como un sistema de ficheros, un controlador de dispositivo o algunas características de la capa superior del núcleo. Un módulo no se ejecuta como su propio proceso o hilo, aunque puede crear los hilos del núcleo que necesite por varios propósitos. En su lugar, un módulo se ejecuta en modo núcleo en nombre del proceso actual.

Por tanto, aunque Linux se puede considerar monolítico, su estructura modular elimina algunas de las dificultades para desarrollar y evolucionar el núcleo. Los módulos cargables de Linux tienen dos características importantes:

- **Enlace dinámico:** un módulo de núcleo puede cargarse y enlazarse al núcleo mientras el núcleo está en memoria y ejecutándose. Un módulo también puede desenlazarse y eliminarse de la memoria en cualquier momento.
- **Módulos apilables:** los módulos se gestionan como una jerarquía. Los módulos individuales sirven como bibliotecas cuando los módulos cliente los referencian desde la parte superior de la jerarquía, y actúan como clientes cuando referencian a módulos de la parte inferior de la jerarquía.

El enlace dinámico facilita la configuración y reduce el uso de la memoria del núcleo. En Linux, un programa de usuario o un usuario puede cargar y descargar explícitamente módulos del núcleo utilizando los mandatos *insmod* y *rmmod*. El núcleo mismo detecta la necesidad de funciones particulares y puede cargar y descargar módulos cuando lo necesite. Con módulos apilables, se pueden definir dependencias entre los módulos. Esto tiene dos ventajas:

1. El código común para un conjunto de módulos similares (por ejemplo, controladores para hardware similar) se puede mover a un único módulo, reduciendo la replicación.
2. El núcleo puede asegurar que los módulos necesarios están presentes, impidiendo descargar un módulo del cual otros módulos que ejecutan dependen y cargando algunos módulos adicionalmente requeridos cuando se carga un nuevo módulo.

Stallings, W. (2006). Introducción a los sistemas operativos. Linux. En *Sistemas operativos* (5ta ed., capítulo 2, pp.96-97). Pearson Education.

5. Intérprete de comandos (Shell)

Ejercicio 5.a

¿Qué es?

> El intérprete de comandos, interfaz de línea de comandos o command line interface (CLI) es el modo de comunicación entre el usuario y el SO. Ejecuta programas a partir del ingreso de comandos.
>
> Una línea de comandos es una interfaz basada en texto que puede utilizarse para introducir instrucciones en un sistema informático. Un programa denominado **shell** proporciona la línea de comandos de Linux.
>
> Los usuarios acceden a la **shell** a través de una **terminal**. Un terminal proporciona un teclado para las entradas del usuario y una pantalla para las salidas. En instalaciones basadas en texto, esta puede ser la **consola** física del equipo Linux, el teclado de hardware y la pantalla. Así, la **consola** se refiere al hardware o la ventana de texto en la que interactúas con el sistema, la **terminal** es la aplicación de software que proporciona esa interfaz de texto, y el **shell** es el programa que interpreta y ejecuta los comandos que ingresas en la terminal.
>
> Red Hat (2017). Acceso a la línea de comandos a través de la consola local. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 1, p. 2). Red Hat.

Ejercicio 5.b

¿Cuáles son sus funciones?

> El shell es la capa más externa del sistema operativo. Los shells incorporan un lenguaje de programación para controlar procesos y archivos, además de iniciar y controlar otros programas. El shell gestiona la interacción entre el usuario y el sistema operativo solicitándole la entrada, interpretando dicha entrada para el sistema operativo y gestionando cualquier resultado de salida procedente del sistema operativo.
>
> Los shells ofrecen un método para comunicarse con el sistema operativo. Esta comunicación tiene lugar de forma interactiva (la entrada desde el teclado se ejecuta inmediatamente) o como un script de shell. Un script de shell es una secuencia de mandatos del shell y del sistema operativo que se almacena en un archivo.
>
> Shells del sistema operativo - IBM

Ejercicio 5.c

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

- **Bourne-Again Shell (bash):** es una versión mejorada de Bourne Shell y es el intérprete de comandos por defecto en la mayoría de las distribuciones de GNU/Linux.
- **Z Shell (zsh):** es un potente intérprete de comandos que puede funcionar como shell interactiva y como intérprete de lenguaje de scripting. En la actualidad es una de las alternativas más populares para Bash.
- **Friendly Interactive Shell (fish):** destaca por su facilidad de uso gracias a su resaltado de sintaxis, autosugerencias, completado de tabulaciones, listas de selección por las que se puede navegar y filtrar y la posibilidad de configurarlo a través de la web.
- **Bourne Shell (sh):** es un intérprete de mandatos interactivo y un lenguaje de programación de mandatos.
- **C Shell (csh):** es un intérprete de mandatos interactivo y un lenguaje de programación de mandatos. Utiliza una sintaxis que es similar al lenguaje de programación C.

Red Hat (2017). Acceso a la línea de comandos a través de la consola local. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 1, p. 2). Red Hat.
Fish Shell
Shells del sistema operativo - IBM

Ejercicio 5.d

¿Dónde se ubican (path) los comandos propios y externos de la Shell?

La ubicación de los **comandos propios** de la Shell varían según el comando para respetar la jerarquía del sistema de archivos. En /bin se encuentran los comandos esenciales para el arranque y funcionamiento básico del sistema. En /usr/bin se encuentran la mayoría de programas y utilidades del sistema. En /sbin y /usr/sbin se encuentran los comandos que solo puede ejecutar el usuario root.

Los **comandos personalizados** pueden estar en cualquier directorio, pero su ruta debe estar indicada en la variable PATH de la Shell. Esta variable puede ser modificada editando el archivo de configuración de la Shell (por ejemplo, .bashrc y .zshrc), que es único para cada usuario.

Ejercicio 5.e

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

Porque el Shell es una aplicación que se ejecuta sobre el kernel y sus comandos se ejecutan en modo usuario, incluso los comandos ejecutados por root.

Ejercicio 5.f

¿Es posible definir un intérprete de comandos distinto para cada usuario? ¿Desde dónde se define? ¿Cualquier usuario puede realizar dicha tarea?

Sí, es posible definir una shell diferente para cada usuario.

La shell de cada usuario particular está definida en el archivo /etc/passwd. En el siguiente ejemplo se puede ver la entrada para el usuario root en /etc/passwd, donde el último campo indica la terminal a utilizar:

```
root:x:0:0:root:/root:/bin/bash
```

Existen tres formas de cambiar la shell de un usuario:

1. **chsh** (change shell) permite que un usuario pueda cambiar su propia shell sin tener privilegios de superusuario (solo tiene que ingresar su contraseña), pero para cambiar la shell de otro usuario si es necesario tener privilegios de superusuario. (Por lo menos en Fedora).

   ```
   $ chsh -s /path/to/shell # cambiar la shell del usuario actual
   # chsh -s /path/to/shell user # cambiar la shell de otro usuario
   ```

2. **usermod** permite modificar las propiedades de una cuenta, almacenados en /etc/passwd. Para modificar el shell se usa la opción -s y siempre se necesitan privilegios de superusuario.

   ```
   # usermod -s /path/to/shell user
   ```

3. Modificar directamente el archivo **/etc/passwd,** para hacerlo se necesitan privilegios de superusuario. Los dos métodos anteriores hacen esto de forma automática.

6. Sistema de Archivos (File System)

Ejercicio 6.a

¿Qué es?

> El sistema de archivos organiza la forma en que se almacenan los archivos en los dispositivos de almacenamiento.
>
> Un sistema de archivos es una estructura organizada de directorios y archivos que contienen datos que residen en un dispositivo de almacenamiento, como una partición o un disco físico. La jerarquía de sistemas de archivos en Linux reúne todos los sistemas de archivos en un árbol de directorios con una sola raíz: el directorio /.
>
> Todos los archivos de un sistema Linux se guardan en sistemas de archivos que están organizados en un árbol de directorios invertido individual conocido como jerarquía de sistema de archivos. Este árbol está invertido porque se dice que la raíz del árbol está en la parte superior de la jerarquía y las ramas de los directorios y subdirectorios se extienden debajo de root.

Red Hat (2017). Jerarquía del sistema de archivos Linux. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 2, p. 30). Red Hat.
Red Hat (2017). Conceptos de la administración del almacenamiento. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 14, p. 378). Red Hat.

Ejercicio 6.b

Mencione sistemas de archivos soportados por GNU/Linux

1. **Ext (Ext2, Ext3, Ext4):** en la actualidad Ext4 es el sistema de archivos por defecto en la mayoría de las distribuciones de GNU/Linux.
2. **Btrfs (B-tree file sistem):** es más moderno que Ext4 y algunas distribuciones como Fedora lo incorporaron como su sistema de archivos por defecto recientemente.
3. **FAT (FAT16, FAT32)**
4. **NTFS:** si bien es un formato propietario de Microsoft, es posible usarlo en Linux mediante herramientas como NTFS-3G, aunque versiones más modernas del kernel ya incluyen los drivers.
5. **etc.**

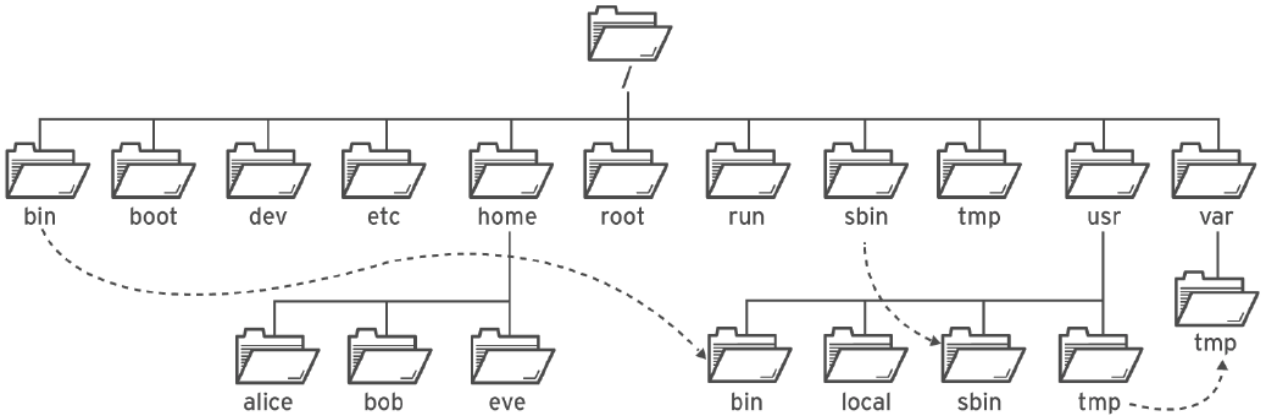
Ejercicio 6.c

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

> Si.

Ejercicio 6.d

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



/	Tope de la estructura de directorios.
/home	Se almacenan archivos de usuarios.
/var	Información que varía de tamaño (logs, BD, spools).
/etc	Archivos de configuración.
/bin	Archivos binarios y ejecutables del sistema.
/dev	Enlace a dispositivos.
/usr	Aplicaciones de usuarios.

Explicación más detallada de todo el FHS:

Ubicación	Descripción
/	Directorio raíz de toda la jerarquía del sistema de archivos.
/bin → /usr/bin	Binarios de comandos esenciales que deben estar disponibles para todos los usuarios (por ejemplo, cat, ls, cp, etc.).
/boot	Contiene archivos estáticos necesarios para arrancar el sistema, como el kernel de Linux. Estos archivos son esenciales para iniciar el proceso de arranque.
/dev	Contiene entradas del sistema de archivos que representan dispositivos conectados al sistema. Estos archivos son esenciales para que el sistema funcione correctamente.

/etc	Este directorio está reservado para archivos de configuración locales de la máquina. No se deben colocar binarios en /etc.
/home	Almacena los directorios de inicio de los usuarios del sistema. Cada usuario tiene su propio subdirectorio dentro de /home/ donde se almacenan sus archivos personales y configuraciones.
/lib → /usr/lib	Solo contiene las bibliotecas necesarias para ejecutar los binarios en /bin/ y /sbin/. Estas imágenes de bibliotecas compartidas son especialmente importantes para arrancar el sistema y ejecutar comandos dentro del sistema de archivos raíz.
/lib64 → /usr/lib64	Son directorios opcionales que suelen utilizarse en sistemas que admiten más de un formato de código ejecutable, como los sistemas que admiten versiones de 32 y 64 bits de un conjunto de instrucciones.
/media	Contiene subdirectorios utilizados como puntos de montaje para dispositivos extraíbles, como unidades USB y CD-ROM.
/mnt	Está reservado para sistemas de archivos montados temporalmente, pero que no son extraíbles (esos van en /media/).
/opt	Proporciona almacenamiento para paquetes de aplicaciones de software estáticos grandes. Un paquete que coloca archivos en el directorio /opt/ crea un directorio con el mismo nombre que el paquete y en él se almacenan archivos que de otro modo estarían distribuidos por todo el sistema de archivos.
/proc	Contiene archivos especiales que extraen o envían información al kernel.
/root	Es el directorio de inicio para root. Es como un /home/ especial para el usuario root.
/run	Datos de tiempo de ejecución para procesos que se iniciaron desde el último arranque. Esto incluye archivos de ID de proceso y archivos de bloqueo, entre otros elementos. El contenido de este directorio se vuelve a crear en el arranque nuevo.
/sbin → /usr/sbin	Almacena los ejecutables utilizados por el usuario root. Los ejecutables de /sbin/ solo se utilizan en el arranque y para realizar operaciones de recuperación del sistema.
/srv	Contiene datos específicos de sitios servidos por el sistema, como datos y scripts para servidores web, datos ofrecidos por servidores FTP y repositorios para sistemas de control de versiones. Los datos que solo pertenecen a un usuario específico deberían ir en el directorio /home/.
/sys	Contiene información sobre dispositivos, drivers y algunas características del kernel.

/tmp	Es un espacio con capacidad de escritura para archivos temporales. Los archivos a los que no se haya accedido, y que no se hayan cambiado ni modificado durante 10 días se eliminan de este directorio automáticamente. Existe otro directorio temporal, /var/tmp, en el que los archivos que no tuvieron acceso, cambios ni modificaciones durante más de 30 días se eliminan automáticamente.
/usr	Software instalado, bibliotecas compartidas, incluye archivos y datos de programa estáticos de solo lectura. Los subdirectorios importantes incluyen: **- /usr/bin:** comandos del usuario. **- /usr/sbin:** comandos de administración del sistema. **- /usr/local:** software personalizado en forma local. **- /usr/etc:** archivos de configuración de todo el sistema. **- /usr/include:** archivos de cabecera para C. **- /usr/lib:** archivos de objetos y bibliotecas que no están diseñados para ser utilizados directamente por usuarios o scripts de shell.
/var	Datos variables específicos de este sistema que deberían conservarse entre los arranques. Los archivos que cambian en forma dinámica (por ejemplo, bases de datos, directorios caché, archivos de registro, documentos en cola de impresión y contenido de sitio web) pueden encontrarse en /var.

El **estándar de jerarquía del sistema de archivos (FHS, Filesystem Hierarchy Standard)** es una norma que define los directorios principales y sus contenidos en el sistema operativo GNU/Linux y otros sistemas de la familia Unix.

Tanto /bin/ como /usr/ contienen binarios, la diferencia entre ambas es que en /bin/ hay binarios del sistema y en /usr/ hay binarios del usuario. Sin embargo, algunas distribuciones de GNU/Linux ponen algunas carpetas del directorio raíz dentro de /usr/ y utilizan un enlace simbólico (una especie de acceso directo) en el directorio raíz para respetar al estándar; tal es el caso de /bin (/usr/bin), /lib (/usr/lib), /lib64 (/usr/lib64) y /sbin (/usr/sbin). (Fedora).

Overview of File System Hierarchy Standard (FHS) - Red Hat

Red Hat (2017). Jerarquía del sistema de archivos Linux. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 2, pp. 30-32). Red Hat.

Filesystem Hierarchy Standard - Wikipedia

¿Cómo se encuentran estructurados los directorios en GNU/Linux? - DesdeLinux

7. Particiones

Ejercicio 7.a

Definición. Tipos de particiones. Ventajas y Desventajas.

Definición:

Los discos duros y los dispositivos de almacenamiento normalmente se dividen en fragmentos más pequeños llamados particiones. Una **partición** es una forma de compartimentar un disco. Sus diferentes partes pueden formatearse con diferentes sistemas de archivos o utilizarse con fines distintos. La colocación de datos en dos sistemas de archivos separados en dos particiones diferentes colabora con la planificación del almacenamiento de datos.

Red Hat (2017). Conceptos de la administración del almacenamiento. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 14, p. 378). Red Hat.

Tipos de particiones:

Ver ejercicio 8.c (esquema de particiones MBR) y ejercicio 8.d (esquema de particiones GPT)

Ventajas y desventajas:

- Cada partición puede formatearse con un tipo de filesystem distrito.
- Es una buena práctica separar los datos del usuario de las aplicaciones y/o el sistema operativo instalado.
- Tener una partición de restauración de todo el sistema.
- Poder ubicar el kernel en una partición de solo lectura, o una que ni siquiera se monta (no está disponible para los usuario).

Algunos ejemplos de situaciones en las cuales la partición del disco es necesaria o beneficiosa son:

- Limitar espacio disponible para aplicaciones o usuarios.
- Permitir varios arranques de diferentes sistemas operativos desde el mismo disco.
- Separar archivos de programa y de sistemas operativos de archivos de usuarios.
- Crear un área separada para el swapping (intercambio) de memoria virtual del sistema operativo.
- Limitar el uso de espacio en disco para mejorar el rendimiento de herramientas de diagnóstico e imágenes de copia de seguridad.

Red Hat (2017). Partición del disco. En *Red Hat System Administration II (RH134-RHEL7-es-3-20170803)* (3ra ed., capítulo 9, p. 178). Red Hat.

Ejercicio 7.b

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

Los dispositivos de almacenamiento se representan con un tipo de archivo especial denominado dispositivo de bloque. El dispositivo de bloque se almacena en el directorio /dev. El primer disco duro SCSI, PATA/SATA o USB detectado es /dev/sda, el segundo es /dev/sdb, y así sucesivamente. Este nombre representa el disco en su totalidad. La primera partición primaria en /dev/sda es /dev/sda1, la segunda es /dev/sda2 y así sucesivamente.

Un listado extenso del archivo de todos los dispositivos /dev/sd* revela que su tipo de archivo especial es **b**, que significa “dispositivo de bloque”:

```
$ ls -l /dev/sd*
brw-rw----. 1 root disk 8, 0 Sep  8 22:29 /dev/sda   # Disco
brw-rw----. 1 root disk 8, 1 Sep  8 22:29 /dev/sda1  # Partición 1
brw-rw----. 1 root disk 8, 2 Sep 10 23:13 /dev/sda2  # Partición 2
```

Nombres de dispositivos en GNU/Linux:

1. **Dispositivos SCSI y SATA:**
   - /dev/sda: Primer disco SCSI o SATA.
   - /dev/sdb: Segundo disco SCSI o SATA.
   - /dev/nvmeXnY: Unidades de estado sólido NVMe, donde X e Y son números que indican el controlador y el número de unidad respectivamente.
2. **Dispositivos IDE (obsoletos):**
   - /dev/hda: Primer disco IDE.
   - /dev/hdb: Segundo disco IDE.
3. **Dispositivos USB y Firewire:**
   - /dev/sdX: Discos USB y Firewire también se nombran como dispositivos SCSI.
4. **Dispositivos MMC/SD (tarjetas SD):**
   - /dev/mmcblk0: Primera tarjeta MMC/SD.
   - /dev/mmcblk0p1: Primera partición en la primera tarjeta MMC/SD.
5. **Dispositivos de CD/DVD:**
   - /dev/sr0 o /dev/scd0: Unidad de CD/DVD ROM.
6. **Dispositivos de cinta (Tape):**
   - /dev/st0: Primer dispositivo de cinta.
7. **Disqueteras**
   - /dev/fd0: Primera disquetera.
8. **RAID (Matrices Redundantes de Discos Independientes):**
   - /dev/mdX: Dispositivos de RAID, donde X es el número de dispositivo de RAID.

9. **Dispositivos virtuales:**
    - /dev/loopX: Dispositivos virtuales utilizados para montar archivos de imagen.
10. **Discos RAM (RAM disks):**
    - /dev/ramX: Discos RAM, donde X es el número de dispositivo.
11. **Máquinas virtuales**
    - /dev/vd<*letter*> o /dev/xvd<*letter*>

Red Hat (2017). Conceptos de la administración del almacenamiento. En *Red Hat System Administration I (RH124-RHEL7-es-3-20170803)* (3ra ed., capítulo 14, p. 378). Red Hat.
ChatGPT.

Ejercicio 7.c

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

Para instalar Linux es necesario como mínimo una partición para el directorio raíz (/), pero es recomendable crear al menos 2 (/ y SWAP). Además, se pueden crear particiones adicionales, por ejemplo para /boot, /home, etc.

Partición	Tipo de partición	Identificación	Sistema de archivos	Punto de montaje
Raíz (/)	Primaria o lógica	No especifica	Ex4, Btrfs, etc.	/
Swap	Primaria o lógica	Swap	No se utiliza (swap)	No se monta directamente
Inicio	Primaria o lógica	No especifica	Ex4, Btrfs, etc.	/boot
Inicio EFI (si se usa UEFI)	Generalmente EFI	EFI	FAT32 (EFI)	/boot/efi

Ejercicio 7.d

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

Existen 3 escenarios posibles:

1. Usar espacio libre no particionado.
2. Usar una partición no usada.
3. Usar espacio libre de una partición activa
    a. Cambio destructivo
    b. Cambio no destructivo

Ejercicio 7.e

¿Qué tipo de software para particionar existe? Menciónelos y compárelos.

- **Destructivos:** Una herramienta de particionamiento destructiva realiza cambios directos en la tabla de particiones o en las estructuras de datos del sistema de archivos. Esto elimina la partición y crea varias más pequeñas en su lugar, de forma tal que cualquier dato presente previamente en la partición original se pierde.
- **No destructivos:** Una herramienta de particionamiento no destructiva intenta realizar operaciones de particionamiento sin afectar negativamente los datos existentes en el disco. Con la repartición no destructiva se ejecuta un programa que hace más pequeña una partición grande sin perder ninguno de los archivos almacenados en esa partición. Este método suele ser fiable, pero puede llevar mucho tiempo en unidades grandes. Consta de tres pasos: (1) compresión y copia de seguridad de los datos existentes, (2) redimensionar la partición existente, (3) crear nueva(s) partición(es).

Algunas de las herramientas más conocidas para particionar discos son:

Nombre	Tipo de interfaz	Esquemas soportados	Tipo de particionado	Descripción
GParted	GUI	MBR, GPT	No destructivo	Permite crear, eliminar, redimensionar y gestionar particiones en discos duros y unidades flash.
KDE Partition Manager	GUI	MBR, GPT	No destructivo	Es similar a GParted y se integra bien con KDE.
parted	CLI	MBR, GPT	No destructivo	Permite gestionar particiones y se utiliza comúnmente en scripts y tareas de administración del sistema.
fdisk	CLI	MBR	Destructivo	Es una herramienta de línea de comandos muy popular
gdisk	CLI	GPT	Destructivo	Es similar a fdisk, pero diseñada específicamente para particionar discos GPT.

Utilizar el espacio libre de una partición activa - Red Hat.

Red Hat (2017). Adición de particiones, sistemas de archivos y montajes persistentes. En *Red Hat System Administration II (RH134-RHEL7-es-3-20170803)* (3ra ed., capítulo 9). Red Hat.

ChatGPT.

8. Arranque (bootstrap) de un sistema operativo

Ejercicio 8.a

¿Qué es el BIOS? ¿Qué tarea realiza?

El **BIOS (Basic Input Output System, Sistema básico de entrada y salida)** es un firmware (un software que maneja físicamente al hardware) que se encuentra en la placa base de una computadora. Está grabado en un chip de solo lectura (ROM) y es el primer programa informático que se ejecuta al encenderlo. Es un software de E/S de bajo nivel que incluye procedimientos para leer el teclado, escribir en la pantalla y realizar operaciones de E/S de disco, entre otras cosas.

Cuando se enciende la computadora, de forma automática (y con ayuda del chipset), la CPU lee el código del BIOS directamente desde la ROM y comienza a ejecutarlo. Lo primero que hace es realizar una serie de verificaciones como parte de las rutinas de pre-arranque de la computadora. Este proceso se conoce como **POST (Power-On Self Test)** y se encarga de realizar las siguientes tareas:

- Verificar los registros de la CPU.
- Verificar la integridad del propio código de la BIOS.
- Verificar algunos componentes básicos como DMA, temporizador y controlador de interrupciones.
- Inicializar, dimensionar y verificar la memoria principal del sistema (RAM).
- Inicializar la BIOS.
- Pasar el control a otras BIOS de extensión especializadas (si están instaladas).
- Identificar, organizar y seleccionar los dispositivos disponibles para el arranque.
- Identificar, inicializar y catalogar todos los buses y dispositivos del sistema.

Si el POST se ejecuta con éxito, entonces el BIOS verifica una lista de dispositivos almacenada en una memoria CMOS (la memoria donde se almacena la configuración de la BIOS) para determinar cuál es el dispositivo de arranque seleccionado. A continuación, lee el primer sector del dispositivo de arranque seleccionado, lo carga en memoria y lo ejecuta.

En este primer sector del dispositivo se encuentra el **sector de arranque maestro (MBR, master boot record)** y se ejecuta un programa que se encuentra al inicio del mismo, conocido como **código maestro de arranque (MBC, Master Boot Code).** El MBC examina la tabla de particiones que se encuentra en la parte final del MBR y determina que partición está marcada como activa. Finalmente, se lee un cargador de arranque secundario dentro de la partición activa (el **bootloader**) y este cargador lee el sistema operativo y lo inicia.

BIOS: ¿Qué es? ¿Para qué sirve en nuestro ordenador? - Profesional Review
Bootloader And Stages of Booting Process Explained! - Embedded Inventor
Power-on self-test - Wikipedia

Tanenbaum, A. S. (2007). Introducción. Arranque de la computadora. En *Sistemas operativos modernos* (3ra ed., capítulo 1, p. 33). Prentice Hall.

Ejercicio 8.b

¿Qué es UEFI? ¿Cuál es su función?

UEFI (Unified Extensible Firmware Interface, Interfaz Unificada de Firmware Extensible) es una especificación que define una interfaz entre el sistema operativo y el firmware y fue desarrollado para solventar muchas de las limitaciones de BIOS, aunque mantiene compatibilidad con él. Tanto la UEFI como la BIOS son software de bajo nivel que se inicia al arrancar el PC, antes de iniciar el sistema operativo, pero la UEFI es una solución más moderna, compatible con discos duros más grandes, tiempos de arranque más rápidos, más funciones de seguridad y, convenientemente, gráficos y cursores de ratón. De hecho, UEFI es esencialmente un sistema operativo pequeño que se ejecuta sobre el firmware del PC, y puede hacer mucho más que una BIOS.

En lugar de almacenar la información sobre la inicialización y el arranque en el firmware, como hace la BIOS, UEFI almacena toda esta información en un archivo .efi, que es almacenado en el disco duro dentro de una partición especial llamada EFI System Partition (ESP, Partición del Sistema EFI), donde también se almacena el o los gestores de arranque (bootloader).

BIOS vs UEFI

Tamaño de discos y particiones

- La BIOS solo puede arrancar desde unidades de 2,1 TB o menos y pueden tener hasta 4 particiones primarias (o 3 particiones primarias y 1 una extendida). Esta limitación se debe a la forma en la que funciona el Master Boot Record.
- El firmware UEFI puede arrancar desde discos de 2,2 TB o más (de hecho, el límite teórico es de 9,4 zettabytes) y puede tener hasta 128 particiones.

Velocidad de arranque

- La BIOS debe ejecutarse en modo procesador de 16 bits, y solo dispone de 1 MB de espacio para ejecutarse. Tiene problemas para inicializar varios dispositivos de hardware a la vez, lo que provoca un proceso de arranque más lento al inicializar todas las interfaces y dispositivos de hardware de un PC moderno.
- UEFI puede ejecutarse en modo de 32 o 64 bits y tiene un mayor espacio de direcciones que BIOS, lo que significa que su proceso de arranque es más rápido. También significa que las pantallas de configuración de UEFI pueden ser más ágiles que las de BIOS, incluidos los gráficos y la compatibilidad con el cursor del ratón.

UEFI sabe cosas

- Todo lo que un firmware de BIOS sabe, en el contexto del arranque del sistema, es qué discos contiene el sistema. El firmware no tiene conocimiento de nada más allá de eso. Ejecuta el gestor de arranque que encuentra en el MBR del disco especificado, y eso es todo. El firmware ya no interviene en el arranque. La capa de firmware no sabe realmente lo que es un gestor de arranque, o lo que es un sistema operativo. Ni siquiera sabe lo que es una partición. Todo lo que puede hacer es ejecutar el gestor de arranque desde el MBR de un disco.
- UEFI proporciona mucha más infraestructura a nivel de firmware para gestionar el arranque del sistema. No es ni de lejos tan simple como BIOS. A diferencia de BIOS, UEFI sí entiende, en mayor o menor medida, los conceptos de “particiones de disco”, “gestores de arranque” y “sistemas operativos”.

MBR vs ESP

- Con MBR, el inicio del disco describe las particiones del disco en un formato particular, y contiene un “cargador de arranque” (bootloader), una pieza muy pequeña de código que el firmware de la BIOS sabe cómo ejecutar y cuyo trabajo es arrancar el sistema o los sistemas operativos. Pero los gestores de arranque modernos suelen ser mucho más grandes de lo que cabe en el espacio del MBR. Para resolver este problema, lo que hacen es instalar una pequeña parte de sí mismos en el MBR propiamente dicho, y el resto en el espacio vacío del disco, entre el final del MBR convencional y el principio de la primera partición (este espacio se conoce como **MBR gap**). El problema de esto es que no hay una convención fiable sobre dónde debe comenzar la primera partición, por lo que es difícil estar seguro de que habrá suficiente espacio.
- UEFI soluciona estas limitaciones del espacio MBR definiendo particiones del sistema EFI (ESP, EFI System Partition). Una ESP es una partición con un formato FAT especial que los firmwares UEFI deben saber leer y se utiliza para almacenar datos y archivos necesarios para el proceso de arranque. El propósito de esto es permitir que todo el mundo confíe en el hecho de que la capa de firmware definitivamente será capaz de leer datos de una partición de disco bastante ‘normal’. La ESP simplifica la gestión de particiones, ya que no es necesario preocuparse por la ubicación exacta de las particiones de arranque. Dentro de la ESP se pueden almacenar ejecutables EFI y los gestores de arranque son ejecutables EFI, así que podríamos tener varios cargadores de arranque para sistemas operativos diferentes en la misma partición sin conflicto, lo que facilita la gestión de múltiples sistemas operativos.

Una explicación bastante detallada: [UEFI boot: how does that actually work, then?]

[UEFI boot: how does that actually work, then?]
[What Is UEFI, and How Is It Different from BIOS? - How-To-Geek]
[UEFI vs BIOS: What’s the difference? - freeCodeCamp]

Ejercicio 8.c

¿Qué es el MBR? ¿Qué es el MBC?

> **Esquema de partición MBR**
> Desde 1982, el esquema de partición de **registro de arranque maestro (MBR, Master Boot Record)** ha dictado cómo los discos se deben particionar en sistemas que ejecutan firmware de BIOS. Este esquema admite un máximo de cuatro particiones principales. En sistemas Linux, con el uso de particiones extendidas y lógicas, se pueden crear un máximo de 15 unidades. Dado que los datos del tamaño de la partición se almacenan como valores de 32 bits, los discos particionados con el esquema MBR tienen un límite de tamaño de disco y partición máximo de 2 TiB. Como consecuencia, de estas limitaciones, el esquema MBR heredado está en proceso de ser sustituido por el nuevo esquema GUID Partition Table (GPT) para la partición de disco.
>
> Red Hat (2017). Partición del disco. En *Red Hat System Administration II (RH134-RHEL7-es-3-20170803)* (3ra ed., capítulo 9, p. 178). Red Hat.

El MBR es una zona reservada en todos los discos físicos que se encuentra en el primer sector del disco (cilindro 0, cabeza 0 y sector 1). El tamaño del MBR coincide con el tamaño estándar de un sector, 512 bytes:

- Los primeros bytes corresponden al **Master Boot Code (MBC).** El MBC es un pequeño código que permite arrancar al SO. La última acción del BIOS es leer el MBC, llevarlo a memoria y ejecutarlo.
- A partir del byte 446 está la **tabla de particiones**, que es de 64 bytes.
- Al final existen 2 bytes libres o para firmar el MBR.

En el esquema de particiones MBR se distinguen entre tres tipos de particiones:

- **Partición primaria:** división cruda del disco (puede haber hasta 4 por disco). Se almacena información de la misma en el MBR.
- **Partición extendida:** sirve para contener unidades lógicas en su interior. Solo puede existir una de partición de este tipo por disco y no se define un tipo de FS directamente sobre ella.
- **Partición lógica:** se crean dentro de la partición extendida y se les define un tipo de FS. Se conectan como una insta enlazada

Debido al limitado tamaño del MBR para guardar el esquema de particiones, en MBR se puede tener hasta un máximo de 4 particiones:

- 4 particiones primarias.
- 3 particiones primarias y una partición extendida.
    - Dentro de la partición extendida puede haber hasta 12 particiones lógicas, dando un total de 15 particiones.

Ejercicio 8.d

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

> **Esquema de partición de GPT**
> Para sistemas que ejecutan firmware Unified Extensible Firmware Interface (UEFI), GPT (GUID Partition Table) es el estándar para el diseño de tablas de partición en discos duros físicos. GPT es parte del estándar UEFI y aborda muchas de las limitaciones impuestas por el antiguo esquema basado en MBR. Según las especificaciones de UEFI, de forma predeterminada, GPT admite hasta 128 particiones. A diferencia de MBR, que usa 32 bits para almacenar información de tamaño y direcciones en bloques lógicos, GPT asigna 64 bits para direcciones en bloques lógicos. Esto asigna GPT para incluir particiones y discos de hasta 8 zebibyte (ZiB), u 8 mil millones de tebibytes (esto es con bloques de 512 bytes, usando discos duros con bloques de 4096 bytes, este límite aumenta a 64 ZiB.).
>
> Red Hat (2017). Partición del disco. En *Red Hat System Administration II (RH134-RHEL7-es-3-20170803)* (3ra ed., capítulo 9, p. 179). Red Hat.

GPT usa un modo de direccionamiento lógico (LBA, logical block addressing) en lugar de cylinder-header-sector. Mantiene un MBR “heredado” para tener compatibilidad con el esquema BIOS que se almacena en el LBA 0. En el LBA 1 está la cabecera GPT. La tabla de particiones en sí está en los bloques sucesivos. La cabecera GPT y la tabla de particiones están escritas al principio y al final del disco para dar redundancia.

Ejercicio 8.e

¿Cuál es la funcionalidad de un “Gestor de Arranque”? ¿Qué tipos existen? ¿Dónde se instalan? Cite gestores de arranque conocidos.

La finalidad del bootloader (gestor de arranque) es la de cargar una imagen de kernel (sistema operativo) de alguna partición para su ejecución. Se ejecuta luego del código del BIOS. Existen dos modos de instalación:

- En el MBR. El bootloader puede llegar a utilizar el MBR gap (el espacio vacío entre el final del MBR y el principio de la primera partición) debido a las limitaciones de espacio del MBR.
- En el sector de arranque de la partición raíz o activa (Volume Boot Record).

La siguiente tabla resume los bootloaders más populares y más importantes:

Bootloader	Descripción
Bootmgr	Bootloader de Microsoft desde Windows Vista y Server 2008.
NTLDR (NT loader)	Bootloader de Microsoft hasta Windows XP y Server 2003.

barebox	Bootloader para sistemas embebidos en impresoras, cámaras, coches, aviones, etc.
boot.efi	Bootloader EFI utilizado desde 2006 en dispositivos Mac.
BootX	Antiguo bootloader de sistemas operativos Mac.
GRUB (Grand Unified Bootloader)	Bootloader libre para sistemas Unix-like como Linux.
ARM Core Bootloader	Bootloader para microcontroladores (usado, por ejemplo, en iPhones)
OpenBIOS	Bootloader libre y portátil bajo licencia GNU-GPL.

What is a bootloader and how does it work? - IONOS

Ejercicio 8.f

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

> Ver ejercicio 8.a (¿Qué es el BIOS?). Los sistemas UEFI funcionan de forma parecida, la gran diferencia es como manejan la carga del SO.

Ejercicio 8.g

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

1. La máquina se enciende. El firmware (BIOS o UEFI) ejecuta una prueba automática de encendido (Power On Self Test, POST), y comienza a inicializar parte del hardware.
2. El firmware del sistema busca un dispositivo de arranque, ya sea configurado en el firmware de arranque UEFI o al buscar un registro de arranque maestro (Master Boot Record, MBR) en todos los discos, en el orden configurado en el BIOS.
3. El firmware del sistema lee un cargador de arranque desde el disco, luego pasa el control del sistema al cargador de arranque. En sistemas GNU/Linux modernos, este será típicamente grub2.
4. El cargador de arranque carga su configuración desde el disco, y presenta al usuario un menú de posibles configuraciones para arrancar.
5. Luego de que el usuario haya hecho una elección (o se haya agotado el tiempo de espera automático), el cargador de arranque carga el kernel y el *initramfs* configurados desde el disco y los coloca en la memoria.

Un *initramfs* es un archivo que contiene módulos del kernel para todo el hardware necesario en el arranque, scripts de inicio y más. En algunas distribuciones de GNU/Linux (especialmente en las empresariales como RHEL o SUSE), initramfs contiene un sistema totalmente utilizable por sí solo. initramfs se utiliza como el primer sistema de archivos raíz al que su máquina tiene acceso. Se utiliza para montar el verdadero rootfs que contiene todos los datos.

6. El cargador de arranque pasa el control del sistema al kernel, y detalla todas las opciones especificadas en la línea de comandos del kernel en el cargador de arranque, y la ubicación del *initramfs* en la memoria.

7. El kernel inicializa todo el hardware para el cual puede encontrar un controlador en el initramfs, y luego ejecuta /sbin/init desde *initramfs* como PID 1.

8. La instancia *systemd* desde *initramfs* ejecuta todas las unidades para el objetivo *initrd.target*. Esto incluye el montaje del sistema de archivos raíz real en /sysroot.

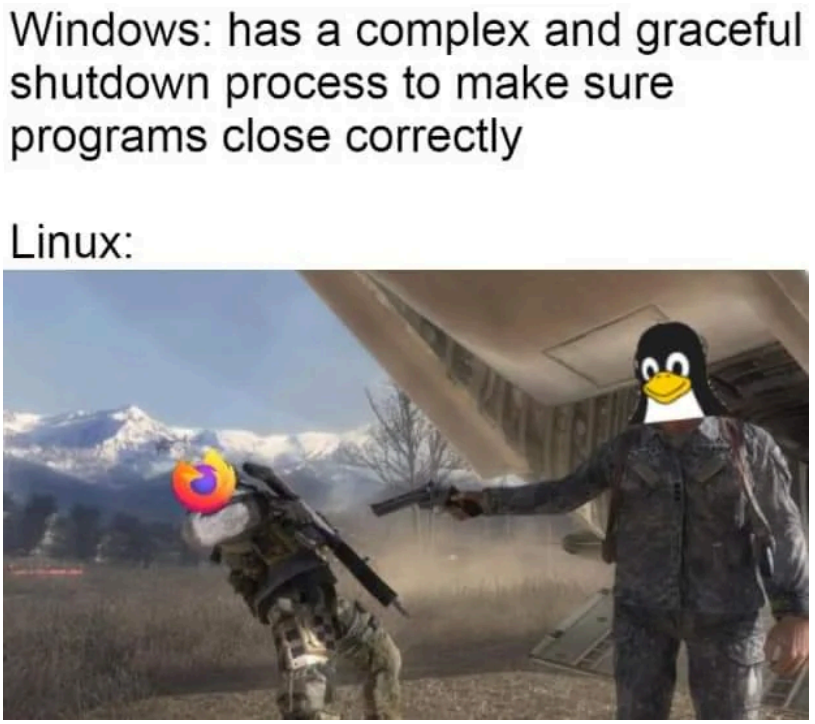
9. El sistema de archivos raíz del kernel se cambia (articula) desde el sistema de archivos raíz de initramfs al sistema de archivos raíz del sistema que se montó anteriormenteen /sysroot. Luego, vuelve a ejecutarse systemd usando la copia de systemd instalado en el sistema.

10. systemd busca un objetivo predeterminado, ya sea especificado desde la línea de comandos del kernel o configurado en el sistema, luego inicia (y detiene) unidades para cumplir con la configuración para ese objetivo, y resuelve dependencias entre unidades automáticamente. En su esencia, un objetivo systemd es un conjunto de unidades que debe activarse para alcanzar un estado de sistema deseado. Estos objetivos incluirán típicamente al menos una pantalla de inicio de sesión basado en texto o inicio de sesión gráfico que se generará.

Red Hat (2017). El Proceso de arranque en RHEL. En *Red Hat System Administration II (RH134-RHEL7-es-3-20170803)* (3ra ed., capítulo 13, pp. 284-285). Red Hat.
Initramfs - Ubuntu Wiki.

Ejercicio 8.h

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



Para apagar o reiniciar un sistema en ejecución desde la línea de comandos, se puede usar el comando systemctl. **systemctl poweroff** detendrá todos los servicios en ejecución, desmontará todos los sistemas de archivos (o volverá a montarlos como solo lectura cuando no se puedan desmontar) y luego apagará el sistema.

Red Hat (2017). Arrancar, Reiniciar y Apagar. En *Red Hat System Administration II (RH134-RHEL7-es-3-20170803)* (3ra ed., capítulo 13, pp. 285). Red Hat.

systemd-poweroff.service es un servicio del sistema que es activado por poweroff.target y es responsable de la operación de apagado del sistema.

Cuando se ejecutan este servicio, se asegura de que el PID 1 sea reemplazado por la herramienta /usr/lib/systemd/systemd-shutdown, que es la responsable del apagado real. Antes de apagarse, este binario intentará desmontar todos los sistemas de archivos restantes (o al menos volver a montarlos como solo lectura), desactivar todos los dispositivos de intercambio restantes, desconectar todos los dispositivos de almacenamiento restantes y matar todos los procesos restantes.

Tenga en cuenta que systemd-poweroff.service (y las unidades relacionadas) nunca deben ejecutarse directamente. En su lugar, active el apagado del sistema con un comando como "systemctl poweroff".

man 8 systemd-poweroff.service

Ejercicio 8.i

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

Sí, es posible tener una PC GNU/Linux y otro sistema operativo instalado a la vez.

En computadoras que utilizan **BIOS** se puede tener dos sistemas operativos diferentes y seleccionar desde el menú de inicio de arranque (boot start menu) desde que dispositivo cargar el sistema de arranque. Pero los sistemas deben estar instalados en discos diferentes, ya que el MBR solo puede guardar un bootloader por disco.

En computadoras con **UEFI**, el firmware está diseñado para reconocer distintos bootloaders en un mismo disco, así que es posible tener más de un sistema instalado en un mismo dispositivo y se puede seleccionar que bootloader cargar desde el menú de arranque.

Además, la mayoría de los **bootloader de GNU/Linux** (como GRUB, LILO, etc.) suelen mostrar un menú donde podemos elegir qué sistema operativo queremos utilizar. Haciendo posible tener más de un sistema operativo en un mismo disco tanto en BIOS como en UEFI.

9. Archivos

Ejercicio 9.a

¿Cómo se identifican los archivos en GNU/Linux?

Linux identifica los archivos por su **File Magic Number;** las extensiones son opcionales, aunque algunos programas la utilizan. Los números mágicos son los primeros bits de un archivo que identifican unívocamente el tipo de archivo. Esto facilita la programación, ya que no es necesario buscar en estructuras de archivo complicadas para identificar el tipo de archivo.

File Magic Numbers - leommoore

Ejercicio 9.b

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

cat encadena archivos y los envía a la salida estándar.

```
cat [OPTION]... [FILE]...
```

more es un filtro para leer texto una pantalla a la vez. Esta versión es especialmente primitiva. Los usuarios deben darse cuenta de que less(1) proporciona una emulación de more(1) además de amplias mejoras.

```
more [options] file...
```

man cat(1) more(1)

Ejercicio 9.c

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

```
$ vi ~/prueba.exe
```

Ejercicio 9.d

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

file es un comando que podemos utilizar para saber el tipo de un archivo.

file prueba cada argumento en un intento de clasificarlo. Hay tres conjuntos de pruebas, realizadas en este orden: pruebas del sistema de archivos, pruebas mágicas y pruebas de lenguaje. La primera prueba que tiene éxito hace que se imprima el tipo de archivo.

Las **pruebas del sistema de archivos** se basan en el examen del resultado de una llamada al sistema stat(2), que comprueba si el fichero está vacío, o si es algún tipo de fichero especial.

Las **pruebas mágicas** se utilizan para comprobar si existen archivos con datos en formatos fijos concretos. Estos archivos tienen un “número mágico” almacenado en un lugar particular cerca del principio del archivo, que indica al sistema operativo UNIX que el archivo es un ejecutable binario, y cuál de varios tipos del mismo. El concepto de “número mágico” se ha aplicado por extensión a los ficheros de datos. Cualquier archivo con algún identificador invariable en un pequeño desplazamiento fijo dentro del archivo se puede describir normalmente de esta manera.

Si un archivo no coincide con ninguna de las entradas del archivo mágico, se examina si parece ser un **archivo de texto**. Las distintas codificaciones de caracteres pueden distinguirse por los diferentes rangos y secuencias de bytes que constituyen el texto

imprimible en cada codificación. Si un archivo supera cualquiera de estas pruebas, se informa de su conjunto de caracteres. Además, file intentará determinar otras características de los archivos de tipo texto. Si las líneas de un fichero terminan con CR, CRLF o NEL, en lugar de con el estándar de Unix LF, se informará de ello. También se identificarán los ficheros que contengan secuencias de escape incrustadas o sobrescrituras.

Una vez que file ha determinado el conjunto de caracteres utilizado en un fichero de tipo texto, intentará determinar en qué **lenguaje** está escrito el fichero. Las pruebas de idioma buscan cadenas concretas (por ejemplo, #include <names.h>) que pueden aparecer en cualquier parte de los primeros bloques de un archivo.

Cualquier archivo que no pueda identificarse como escrito en ninguno de los conjuntos de caracteres enumerados anteriormente se considera simplemente data (datos).

man 1 file

10. Comandos I

Indique qué comando es necesario utilizar para realizar cada una de las siguientes acciones. Investigue su funcionamiento y parámetros más importantes:

```
a. Cree la carpeta ISO2017
iso@debian-vm:~$ mkdir ISO2017

b. Acceda a la carpeta
iso@debian-vm:~$ cd ISO2017/

c. Cree dos archivos con los nombres iso2017-1 iso2017-2 (touch)
iso@debian-vm:~/ISO2017$ touch iso2017-1 iso2017-2

d. Liste el contenido del directorio actual
iso@debian-vm:~/ISO2017$ ls
iso2017-1  iso2017-2

d. Visualizar la ruta donde estoy situado (pwd)
iso@debian-vm:~/ISO2017$ pwd
/home/iso/ISO2017

f. Busque todos los archivos en los que su nombre contiene la cadena
“iso*” (find)
iso@debian-vm:~/ISO2017$ sudo find / -name "iso*"
/boot/grub/i386-pc/iso9660.mod
/sys/devices/system/cpu/isolated
/home/iso
/home/iso/ISO2017/iso2017-1
/home/iso/ISO2017/iso2017-2
...

g. Informar la cantidad de espacio libre en disco (df)
iso@debian-vm:~/ISO2017$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G     0  1.9G   0% /dev
tmpfs           392M  1.3M  391M   1% /run
/dev/vda1       6.8G  4.6G  1.9G  71% /
tmpfs           2.0G     0  2.0G   0% /dev/shm
tmpfs           5.0M  8.0K  5.0M   1% /run/lock
/dev/vda6        12G  121M   12G   2% /home
tmpfs           392M   92K  392M   1% /run/user/1000
```

```
h. Verifique los usuarios conectados al sistema (who)
iso@debian-vm:~/ISO2017$ who
iso      tty2         2023-09-15 20:54 (tty2)
iso@debian-vm:~/ISO2017$ sudo who
iso      tty2         2023-09-15 20:54 (tty2)
iso      pts/2        2023-09-15 21:19

i. Acceder al archivo iso2017-1 e ingresar Nombre y Apellido
iso@debian-vm:~/ISO2017$ vi iso2017-1

j. Mostrar en pantalla las últimas líneas de un archivo (tail).
iso@debian-vm:~/ISO2017$ tail -3 /etc/passwd
gnome-initial-setup:x:112:65534::/run/gnome-initial-setup/:/bin/false
Debian-gdm:x:113:121:Gnome Display Manager:/var/lib/gdm3:/bin/false
iso:x:1000:1000:iso,,,:/home/iso:/bin/bash
```

11. Comandos II

Investigue su funcionamiento y parámetros más importantes:

`shutdown [OPTIONS...] [TIME] [WALL...]`

shutdown may be used to halt, power off, or reboot the machine.

The first argument may be a time string (which is usually "now"). Optionally, this may be followed by a wall message to be sent to all logged-in users before going down.

-H, --halt	Halt the machine.
-P, --poweroff	Power the machine off (the default).
-r, --reboot	Reboot the machine.
-H, --halt	Halt the machine.

`poweroff [OPTIONS...]; reboot [OPTIONS...]; halt [OPTIONS...]`

poweroff, **reboot**, and **halt** may be used to power off, reboot, or halt the machine. All three commands take the same options.

--halt	Halt the machine, regardless of which one of the three commands is invoked.
-p, --poweroff	Power off the machine, when either **halt** or **poweroff** is invoked. This option is ignored when **reboot** is invoked.
--reboot	Reboot the machine, regardless of which one of the three commands is invoked.
-f, --force	Force immediate power-off, halt, or reboot. If specified, the command does not contact the init system. In most cases, filesystems are not properly unmounted before shutdown.

`plocate [OPTION]...  PATTERN...`

plocate *(a faster version of locate)* finds all files on the system matching the given pattern (or all of the patterns if multiple are given). It does this by means of an index made by updatedb(8) or (less commonly) converted from another index by plocate-build(8).

When multiple patterns are given, plocate will search for files that match all of them. By default, patterns are taken to be substrings to search for. If at least one non-escaped globbing metacharacter (*, ? or []) is given, that pattern is instead taken to be a glob pattern (which means it needs to start and end in * for a substring match).

Before locate(1) can be used, the database will need to be created, this is done with the updatedb(8) command, which (as the name suggests) updates the database.

-b, --basename	Match only against the file name portion of the path name, ie., the directory names will be excluded from the match (but still printed). This does not speed up the search, but can suppress uninteresting matches.
-c, --count	Do not print each match. Instead, count them, and print out a total number at the end.
-i, --ignore-case	Do a case-insensitive match as given by the current locale (default is case-sensitive, byte-by-byte match).
-l, --limit *LIMIT*	Stop searching after LIMIT matches have been found. If --count is given, the number printed out will be at most LIMIT.

`uname [OPTION]...`

Print certain system information. With no OPTION, same as -s.

-a, --all	Print all information, in the following order, except omit -p and -i if unknown.
-s, --kernel-name	Print the kernel name
-n, --nodename	Print the network node hostname
-r, --kernel-release	Print the kernel release
-v, --kernel-version	Print the kernel version
-m, --machine	Print the machine hardware name
-p, --processor	Print the processor type (non-portable)
-i, -hardware-platform	Print the hardware platform (non-portable)
-o, -operating-system	Print the operating system

`lspci [options]`

lspci list all PCI devices. It's a utility for displaying information about PCI buses in the system and devices connected to them. Some parts of the output, especially in the highly verbose modes, are probably intelligible only to experienced PCI hackers.

Access to some parts of the PCI configuration space is restricted to root on many operating systems, so the features of lspci available to normal users are limited.

-v	Be verbose and display detailed information about all devices.
-vv	Be very verbose and display more details. This level includes everything deemed useful.
-vvv	Be even more verbose and display everything we are able to parse, even if it doesn't look interesting at all (e.g., undefined memory regions).
-k	Show kernel drivers handling each device and also kernel modules capable of handling it. Turned on by default when -v is given in the normal mode of output. (Currently works only on Linux with kernel 2.6 or newer.)
-x	Show hexadecimal dump of the standard part of the configuration space (the first 64 bytes or 128 bytes for CardBus bridges).

```
at [-V] [-q queue] [-f file] [-u username] [-mMlv] timespec ...
at [-V] [-q queue] [-f file] [-u username] [-mMkv] [-t time]
atq [-V] [-q queue] [-o timeformat] [job ...]
atrm [-V] job [...]
batch
```

at, batch, atq, atrm - queue, examine, or delete jobs for later execution**. at** and **batch** read commands from standard input or a specified file which are to be executed at a later time, using /bin/sh.

at	Executes commands at a specified time.
atq	Lists the user's pending jobs, unless the user is the superuser; in that case, everybody's jobs are listed. The format of the output lines (one for each job) is: Job number, date, hour, queue, and username.
atrm	Deletes jobs, identified by their job number.
batch	Executes commands when system load levels permit; in other words, when the load average drops below 1.5, or the value specified in the invocation of atd.

If you specify a job to absolutely run at a specific time and date in the past, the job will run as soon as possible. For example, if it is 8pm, and you do at 6pm today, it will run more likely at 8:05pm.

The definition of the time specification can be found in /usr/share/doc/at/timespec.

The superuser may use these commands in any case. For other users, permission to use at is determined by the files /etc/at.allow and /etc/at.deny. See at.allow(5) for details.

-f *file*	Reads the job from file rather than standard input.

-t *time*	Run the job at time, given in the format [[CC]YY]MMDDhhmm[.ss].
-v	Shows the time the job will be executed before reading the job.
-c	Cats the jobs listed on the command line to standard output.
-o *fmt*	strftime-like time format used for the job list.

```
netstat [-AaLlnW] [-f address_family | -p protocol] [-M core]
        [-N system]
netstat [-gilns] [-f address_family] [-M core] [-N system]
netstat -i | -I interface [-w wait] [-abdgt] [-M core] [-N system]
netstat -s [-s] [-f address_family | -p protocol] [-M core]
        [-N system]
netstat -i | -I interface -s [-f address_family | -p protocol]
        [-M core] [-N system]
netstat -m [-M core] [-N system]
netstat -r [-Aaln] [-f address_family] [-M core] [-N system]
netstat -rs [-s] [-M core] [-N system]
```

The **netstat** command symbolically displays the contents of various network-related data structures. There are a number of output formats, depending on the options for the information presented. The ***first*** form of the command displays a list of active sockets for each protocol. The ***second*** form presents the contents of one of the other network data structures according to the option selected. Using the ***third*** form, with a wait interval specified, **netstat** will continuously display the information regarding packet traffic on the configured network interfaces. The ***fourth*** form displays statistics for the specified protocol or address family. The ***fifth*** form displays per-interface statistics for the specified protocol or address family. The ***sixth*** form displays mbuf(9) statistics. The ***seventh*** form displays the routing table for the specified address family. The ***eighth*** form displays routing statistics.

```
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 serves to attach the filesystem found on some device to the big file tree. Conversely, **umount** will detach it again. The filesystem is used to control how data is stored on the device or provided in a virtual way by network or other services.

The standard form of the mount command is: `mount -t type device dir`

This tells the kernel to attach the filesystem found on device (which is of type *type*) at the directory dir. The option -t type is optional. The mount command is usually able to detect

a filesystem. The root permissions are necessary to mount a filesystem by default. The previous contents (if any) and owner and mode of dir become invisible, and as long as this filesystem remains mounted, the pathname *dir* refers to the root of the filesystem on device.

-a, --all	Mount all filesystems (of the given types) mentioned in ***fstab*** (except for those whose line contains the ***noauto*** keyword). The filesystems are mounted following their order in ***fstab***.
-B, --bind	Remount a subtree somewhere else (so that its contents are available in both places).
-R, --rbind	Remount a subtree and all possible submounts somewhere else (so that its contents are available in both places).
-M --move	Move a subtree to some other place. See above, the subsection The move operation.
-o, --options *opts*	Use the specified **mount** options. The ***opts*** argument is a comma-separated list.
-r, --read-only	Mount the filesystem read-only. A synonym is -o ro.
-w, --rw, --read-write	Mount the filesystem read/write. A synonym is -o rw.
-t, --types *fstype*	***fstype*** is used to indicate the filesystem type. The filesystem types which are currently supported depend on the running kernel. If no -t option is given, or if the auto type is specified, mount will try to guess the desired type. The prefix no can be meaningful with the **-a** option. For example, the command `mount -a -t nomsdos,smbfs` mounts all filesystems except those of type *msdos* and *smbfs*.
-U, --uuid *uuid*	Mount the partition that has the specified uuid.

```
umount -a [-dflnrv] [-t fstype] [-O option...]
umount [-dflnrv] {directory|device}
umount -h|-V
```

The umount command detaches the mentioned filesystem(s) from the file hierarchy. A filesystem is specified by giving the directory where it has been mounted. Giving the special device on which the filesystem lives may also work, but is obsolete, mainly because it will fail in case this device was mounted on more than one directory.

-a, --all	All of the filesystems described in /proc/self/mountinfo are unmounted.

-r, --read-only When an unmount fails, try to remount the filesystem read-only.

`head [OPTION]... [FILE]...`

Print the first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input

-n, --lines=[-]NUM Print the first NUM lines instead of the first 10; with the leading '-', print all but the last NUM lines of each file.

`losetup`

losetup is used to associate loop devices with regular files or block devices, to detach loop devices, and to query the status of a loop device. If only the ***loopdev*** argument is given, the status of the corresponding loop device is shown. If no option is given, all loop devices are shown.

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] [-Pr] [--show] -f|loopdev file

Resize a loop device:
losetup -c loopdev

`write user [ttyname]`

write allows you to communicate with other users. When you run the write command, the user you are writing to gets a message of the form:
Message from *yourname*@*yourhost* on *yourtty* at *hh*:*mm* ...

Any further lines you enter will be copied to the specified user's terminal. If the other user wants to reply, they must run write as well. When you are done, type an end-of-file or interrupt character. The other user will see the message EOF indicating that the conversation is over.

You can prevent people (other than the superuser) from writing to you with the mesg(1) command. Some commands, for example nroff(1) and pr(1), may automatically disallow writing, so that the output they produce isn't overwritten.

If the user you want to write to is logged in on more than one terminal, you can specify which terminal to write to by giving the terminal name as the second operand to the write command. Alternatively, you can let write select one of the terminals - it will pick the one with the shortest idle time. This is so that if the user is logged in at work and also dialed up from home, the message will go to the right place.

The traditional protocol for writing to someone is that the string -o, either at the end of a line or on a line by itself, means that it's the other person's turn to talk. The string oo means that the person believes the conversation to be over.

`mkfs [options] [-t type] [fs-options] device [size]`

mkfs is used to build a Linux filesystem on a device, usually a hard disk partition. The device argument is either the device name (e.g., /dev/hda1, /dev/sdb2), or a regular file that shall contain the filesystem. The size argument is the number of blocks to be used for the filesystem.

This mkfs frontend is deprecated in favour of filesystem specific mkfs.<type> utils. In actuality, **mkfs** is simply a front-end for the various filesystem builders (mkfs.fstype) available under Linux. The filesystem-specific builder is searched for via your PATH environment setting only.

-t, --type *type*	Specify the type of filesystem to be built. If not specified, the default filesystem type (currently ext2) is used.
fs-options	Filesystem-specific options to be passed to the real filesystem builder.

---

fdisk [options] device
fdisk -l [device...]

fdisk is a dialog-driven program for creation and manipulation of partition tables. It understands GPT, MBR, Sun, SGI and BSD partition tables.

All partitioning is driven by device I/O limits (the topology) by default. **fdisk** is able to optimize the disk layout for a 4K-sector size and use an alignment offset on modern devices for MBR and GPT. It is always a good idea to follow fdisk's defaults as the default values (e.g., first and last partition sectors) and partition sizes specified by the +/-<size>{M,G,...} notation are always aligned according to the device properties.

---

man pages en Fedora Linux 38

12. Comandos III

Indique en qué directorios se almacenan los comandos mencionados en el ejercicio anterior.

```
#!/bin/bash

# Recorrer todos los argumentos (comandos) recibidos
for cmd in "$@"; do
    # Utilizar 'which' para encontrar la ubicación del comando y
    # descartar el stderr para mostrar un mensaje más lindo :3
    cmd_location=$(which "$cmd" 2> /dev/null)

    # Verificar si el comando existe y tiene una ubicación
    if [[ $? -eq 0 ]]; then
      printf '%-10s %s\n' $cmd $cmd_location
    else
      printf '%-10s no se encuentra en el sistema.\n' $cmd
    fi
done
```

ubicacionComandos.sh

```
iso@debian-vm:~$ bash ubicacionComandos.sh shutdown reboot halt locate
uname gmesg lspci at netstat mount umount head losetup write mkfs fdisk

shutdown   /usr/sbin/shutdown
reboot     /usr/sbin/reboot
halt       /usr/sbin/halt
locate     /usr/bin/locate
uname      /usr/bin/uname
gmesg      no se encuentra en el sistema.
lspci      /usr/sbin/lspci
at         /usr/bin/at
netstat    /usr/bin/netstat
mount      /usr/bin/mount
umount     /usr/bin/umount
head       /usr/bin/head
losetup    /usr/sbin/losetup
write      /usr/bin/write
mkfs       /usr/sbin/mkfs
fdisk      /usr/sbin/fdisk
```