

1) Características de GNU/Linux:

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

- Es un sistema operativo Unix-like
- Gratuito y de libre distribución
- Es código abierto
- Rápida corrección ante fallas

Además, GNU se refiere a 4 libertades principales:

- Libertad de usar el programa con cualquier propósito
- Libertad de estudiar su funcionamiento
- Libertad para distribuir sus copias
- Libertad para mejorar los programas

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

Windows y Mac Os son sistemas operativos de propietario, es decir, una empresa administra el código y su distribución.

Estos son de paga y no poseen código abierto, por lo tanto no es modificable, estudiable, etc.

Toda corrección de fallas queda en manos del propietario, por lo tanto hay menor necesidad de técnicos especializados.

No todos los sistemas operativos de propietario son de pago, usualmente lo son. Misma comparativa con el software libre, no quiere decir que sea gratuito.

c) ¿Qué es GNU?

Las siglas de GNU significan “GNU’s not Unix” (GNU no es Unix).

Es una colección de paquetes de software que pueden ser utilizados como sistema operativo o adaptarse a otro sistema operativo.

La mayoría del software de GNU está licenciado bajo su propia GPL (General Public License), lo que permite que sea estudiado, distribuido y modificado gratuita y libremente.

d) Indique una breve historia sobre la evolución del proyecto GNU.

\*Robado de <https://github.com/agusrnfr>\*

Fue iniciado por Richard Stallman en 1983 con el fin de crear un Unix libre (el sistema GNU). En 1985 Richard Stallman creó la Free Software Foundation (FSF o Fundación para el Software Libre) para proveer soportes logísticos, legales y financieros al proyecto GNU. Esta fundación contrato programadores,

aunque una porción sustancial del desarrollo fue (y continúa siendo) producida por voluntarios.

En 1990, GNU ya contaba con un editor de textos (Emacs), un compilador (GCC) y gran cantidad de bibliotecas que componen un Unix típico pero faltaba el componente principal, el núcleo (Kernel).

Linus Torvalds ya venía trabajando desde 1991 en un Kernel denominado Linux, el cual se distribuía bajo licencia GPL. Múltiples programadores se unieron a Linus en el desarrollo, colaborando a través de Internet y consiguiendo paulatinamente que Linux llegase a ser un núcleo compatible con UNIX. En 1992, el núcleo Linux fue combinado con el sistema GNU, resultando en un SO libre y completamente funcional. El SO formado por esta combinación es usualmente conocido como "GNU/Linux" o como una "distribución Linux" y existen diversas variantes.

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

GNU/Linux es multiusuario, multitarea y multiprocesador:

Multitarea: Es la capacidad de un sistema operativo para ejecutar múltiples tareas o procesos al mismo tiempo. El CPU generalmente ejecuta solo una instrucción de una tarea a la vez, pero los sistemas operativos modernos pueden alternar entre tareas rápidamente para parece que se realizan en simultáneo.

Esto se logra mediante la planificación de procesos:

- Preemptive Multitasking: El sistema operativo decide cuándo interrumpir una tarea para dar tiempo de CPU a otra tarea, mejorando la capacidad de respuesta del sistema.
- Cooperative Multitasking: Las tareas ceden voluntariamente el control de la CPU a otras tareas.

Multiusuario: Capacidad de un sistema operativo para permitir que varios usuarios utilicen la misma computadora o recursos de la red al mismo tiempo o en diferentes momentos. (Ejemplo: un servidor).

Cada usuario tiene su propio espacio de trabajo, archivos y permisos. Los sistemas operativos que soportan multiusuario gestionan la seguridad y los permisos para garantizar que un usuario no pueda interferir con el trabajo de otro.

Multiprocesador: Se refiere a un sistema con más de un procesador (CPU) que puede ejecutar múltiples procesos simultáneamente.

Los sistemas multiprocesador pueden distribuir tareas entre los diferentes procesadores, mejorando significativamente el rendimiento, especialmente para tareas intensivas en procesamiento. El sistema operativo debe soportar el procesamiento paralelo:

- SMP (Symmetric Multiprocessing): Todos los procesadores comparten un mismo espacio de memoria y son iguales en términos de acceso a los recursos.
- AMP (Asymmetric Multiprocessing): Un procesador principal controla el sistema y asigna tareas a los procesadores secundarios.

f) ¿Qué es el POSIX?

\*Robado de <https://github.com/agusrnfr>\*

POSIX consiste en una familia de estándares especificadas por la IEEE con el objetivo de facilitar la interoperabilidad de sistemas operativos. Además, POSIX establece las reglas para la portabilidad de programas. Por ejemplo, cuando se desarrolla software que cumple con los estándares POSIX existe una gran probabilidad de que se podrá utilizar en sistemas operativos del tipo Unix. Si se ignoran tales reglas, es muy posible que el programa o librería funcione bien en un sistema dado pero que no lo haga en otro.

## 2) Distribuciones de GNU/Linux:

a) ¿Qué es una distribución de GNU/Linux? Nombre al menos 5 distribuciones de GNU/Linux y cite diferencias básicas entre ellas

Una distribución es una customización de GNU/Linux formada por una versión de Kernel y un conjunto de herramientas y software adicionales, configurados de manera específica para cumplir con diferentes propósitos o satisfacer diferentes necesidades de los usuarios.

Cada distribución tiene características propias y es tarea del usuario elegir una acorde a sus necesidades y gustos personales.

\*Robado de <https://github.com/agusrnfr>\*

Distribuciones como **Ubuntu** se centran en ser lo más amigables posible a la hora de instalarse o descargar programas. **Linux Mint** aprovecha el hardware potente para competir con Windows o MacOS. Para computadoras viejas hay distribuciones ligeras como **Puppy Linux**. Para Linux en servidores, **Debian**, y para jugar videojuegos, **SteamOS** es la mejor.

Hay distribuciones que desarrollan compañías comerciales, como **Fedora**, **Mandriva** o la propia Ubuntu, y otras mantenidas por la comunidad Linux, como Debian, que no está relacionada con ninguna empresa y utiliza únicamente software.

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

Las distribuciones de Linux varían en varios aspectos, lo que les permite satisfacer diferentes necesidades, preferencias y niveles de experiencia.

- Gestor de paquetes: Cada distribución utiliza un gestor de paquetes específico para instalar, actualizar y gestionar Software:

**Debian, Ubuntu, Linux Mint** utilizan 'APT' (Advanced Package Tool) con paquetes '.deb'.

**Fedora, CentOS, RHEL** utilizan 'DNF' (anteriormente 'YUM') con paquetes '.rpm'.

**Arch Linux** utiliza 'Pacman' con paquetes '.pkg.tar.zst'.

**openSuse** utiliza 'Zypper' con paquetes '.rpm'.

- Filosofía y propósito: Las distribuciones tienen diferentes enfoques y objetivos:

**Debian** enfatiza la estabilidad y software libre.

**Ubuntu** deriva de Debian, se enfoca en la facilidad de uso.

**Fedora** patrocinada por Red Hat, se centra en incorporar las últimas tecnologías.

**Arch Linux** ofrece un enfoque minimalista, diseñado para usuario avanzados que prefieren construir su sistema desde cero.

**CentOS/RHEL** orientadas a servidores y empresas, se centran en la estabilidad a largo plazo y soporte comercial.

- Entornos de Escritorio: Aunque el entorno se puede customizar en casi cualquier distribución, algunas prefieren ciertos entornos:

**Ubuntu** viene con GNOME por defecto.

**Linux Mint** ofrece Cinnamon, MATE y Xfce como entornos principales.

**Fedora** por defecto incluye GNOME, pero ofrece ediciones con entornos como KDE, Plasma y Xfce.

- Soporte y Ciclo de Vida: El tiempo de soporte y la frecuencia de actualizaciones varía:

**Ubuntu LTS (Long Term Support)** ofrece 5 años de soporte en sus versiones LTS.

**Fedora** tiene un ciclo de vida de ~13 meses, con versiones nuevas lanzadas cada 6 meses.

**Arch Linux** es una distribución "rolling release", lo que significa que se actualiza continuamente en lugar de tener versiones periódicas.

**Debian Stable** tiene un ciclo de lanzamiento de aproximadamente 2 años, priorizando la estabilidad.

- Facilidad de Uso: Las distribuciones varían en su complejidad y la curva de aprendizaje necesaria:

**Ubuntu y Linux Mint** están diseñadas para ser amigables para principiantes, con instaladores gráficos y configuraciones predeterminadas fáciles de usar.

**Debian** es robusta y estable, pero su instalación puede ser más compleja.

**Arch Linux** requiere conocimientos avanzados debido a que la instalación y configuración inicial son completamente manuales.

**Fedora** es un punto intermedio.

Entre muchas otras diferencias.

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

\*Robado de <https://github.com/agusrnfr>\*

El Proyecto Debian es una asociación de personas que han hecho causa común para crear un sistema operativo libre.

Es producido por alrededor de un millar de desarrolladores activos, dispersos por el mundo que ayudan voluntariamente en su tiempo libre. La comunicación se realiza principalmente mediante correo electrónico (listas de correo en [lists.debian.org](mailto:lists.debian.org)) y de IRC (canal [#debian](irc://irc.debian.org) [irc.debian.org](irc://irc.debian.org)).

Comenzó en agosto de 1993 gracias a Ian Murdock, como una nueva distribución que se realizaría de forma abierta, en la línea del espíritu de Linux y GNU. Debian estaba pensado para ser creada de forma cuidadosa y concienzuda, y ser mantenida y soportada con el mismo cuidado. Comenzó como un grupo de pocos y fuertemente unidos hackers de Software Libre, y gradualmente creció hasta convertirse en una comunidad grande y bien organizada de desarrolladores y usuarios.

### 3) Estructura de GNU/Linux:

- a) Nombres cuales son los 3 componentes fundamentales de GNU/Linux.

- Kernel (Núcleo)
- Shell (Intérprete de comandos)
- FileSystem (Sistema de archivos)

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

El Kernel 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 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.

La Shell o también conocida como CLI (Command Line Interface) es el modo de comunicación entre el usuario y el sistema operativo.

El FileSystem organiza la forma en que se almacenan los archivos en dispositivos de almacenamiento (fat, ntfs, ext2, ext3, reiser, etc).

El adoptado por GNU/Linux es el Extended (ext2, ext3, etc).

Los directorios más importantes según el FHS (FileSystem Hierarchy Standard) son:

- / Tope de la estructura de directorios (Similar al C:\ en Windows)
- /home Se almacenan archivos de usuarios (Mis Documentos)
- /var Información que varía de tamaño (Logs, DB, spools)
- /etc Archivos de configuración
- /bin Archivos binarios y ejecutables
- /dev Enlace a dispositivos
- /usr Aplicaciones de usuarios

#### 4) Kernel:

- a) ¿Qué es? INdique una breve reseña histórica acerca de la evolución del Kernel de GNU/Linux.

\*Robado de <https://github.com/agusrnfr>\*

El kernel (también conocido como núcleo) es la parte fundamental de un sistema operativo. El kernel o núcleo de linux se podría definir como el corazón de este sistema operativo. Es, a grandes rasgos, el encargado de que el software y el hardware de una computadora puedan trabajar juntos.

Historia:


- En 1991 Linus Torvalds inicia la programación de un Kernel Linux basado en Minix.
- El 5 de octubre de 1991, se anuncia la primera versión "oficial" de Linux (0.02)
- En 1992 se combina su desarrollo con GNU, formando GNU/Linux
- La versión 1.0 apareció el 14 de marzo de 1994
- En mayo de 1996 se decide adoptar a Tux como mascota oficial de Linux
- En julio de 1996 se lanza la versión 2.0 y se define la nomenclatura de versionado<sup>1</sup>. Se desarrollo hasta febrero de 2004 y termino con la 2.0.40
- En enero de 1999 se lanza la versión 2.2, que provee mejoras de portabilidad entre otras y se desarrolla hasta febrero de 2004 terminando en la versión 2.2.26
- En 2001 se lanza la versión 2.4 y se deja de desarrollar a fines del 2010 con la 2.4.37.11
- La versión 2.4 fue la que catapulto a GNU/Linux como un SO estable y robusto. Durante este periodo es que se comienza a utilizar Linux más asiduamente
- A fines del 2003 se lanza la versión 2.6. Esta versión ha tenido muchas mejoras para el SO dentro de las que se destacan soporte de hilos, mejoras en la planificación y soporte de nuevo hardware
- El 3 de agosto de 2011 se lanza la versión 2.6.39.4 anunciándose la misma desde meses previos como la última en su revisión
- El 17 de julio de 2011 se lanza la versión 3.01 por los 20 años del SO. Es totalmente compatible con 2.6
- La última versión estable es la 4.7.1 (agosto de 2016)

b) ¿Cuáles son sus funciones principales?

Sus funciones más importantes son la administración de memoria, CPU y E/S.

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

Protocol	Location
HTTP	<a href="https://www.kernel.org/pub/">https://www.kernel.org/pub/</a>
GIT	<a href="https://git.kernel.org/">https://git.kernel.org/</a>
RSYNC	<a href="rsync://rsync.kernel.org/pub/">rsync://rsync.kernel.org/pub/</a>

Latest Release  
**6.10.6** 

mainline:	6.11-rc4	2024-08-18	<a href="#">[tarball]</a>	<a href="#">[patch]</a>	<a href="#">[inc. patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a>	
stable:	6.10.6	2024-08-19	<a href="#">[tarball]</a>	<a href="#">[pgp]</a>	<a href="#">[patch]</a>	<a href="#">[inc. patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a> <a href="#">[changelog]</a>
stable:	6.9.12 [EOL]	2024-07-27	<a href="#">[tarball]</a>	<a href="#">[pgp]</a>	<a href="#">[patch]</a>	<a href="#">[inc. patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a> <a href="#">[changelog]</a>
longterm:	6.6.47	2024-08-19	<a href="#">[tarball]</a>	<a href="#">[pgp]</a>	<a href="#">[patch]</a>	<a href="#">[inc. patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a> <a href="#">[changelog]</a>
longterm:	6.1.106	2024-08-19	<a href="#">[tarball]</a>	<a href="#">[pgp]</a>	<a href="#">[patch]</a>	<a href="#">[inc. patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a> <a href="#">[changelog]</a>
longterm:	5.15.165	2024-08-19	<a href="#">[tarball]</a>	<a href="#">[pgp]</a>	<a href="#">[patch]</a>	<a href="#">[inc. patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a> <a href="#">[changelog]</a>
longterm:	5.10.224	2024-08-19	<a href="#">[tarball]</a>	<a href="#">[pgp]</a>	<a href="#">[patch]</a>	<a href="#">[inc. patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a> <a href="#">[changelog]</a>
longterm:	5.4.282	2024-08-19	<a href="#">[tarball]</a>	<a href="#">[pgp]</a>	<a href="#">[patch]</a>	<a href="#">[inc. patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a> <a href="#">[changelog]</a>
longterm:	4.19.320	2024-08-19	<a href="#">[tarball]</a>	<a href="#">[pgp]</a>	<a href="#">[patch]</a>	<a href="#">[inc. patch]</a>	<a href="#">[view diff]</a>	<a href="#">[browse]</a> <a href="#">[changelog]</a>
linux-next:	next-20240823	2024-08-23						<a href="#">[browse]</a>

*Última versión disponible 6.10.6 (24 de agosto 2024)*

\*Robado de <https://github.com/agusrnfr>\*

Antes de la serie de Linux 2.6.x, los números pares indicaban la versión “estable” lanzada. Por ejemplo, una para uso de fabricación, como el 1.2, 2.4 ó 2.6. Los números impares, en cambio, como la serie 2.5.x, son versiones de desarrollo, es decir que no son consideradas de producción.

Comenzando con la serie Linux 2.6.x, no hay gran diferencia entre los números pares o impares con respecto a las nuevas herramientas desarrolladas en la misma serie del kernel. Linus Torvalds dictaminó que este será el modelo en el futuro.

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

Sí, es posible tener más de un Kernel instalado a la vez, pero esto no significa que se puedan usar en simultáneo.

Muchas distribuciones de Linux permiten tener múltiples versiones del Kernel instaladas al mismo tiempo, esto se gestiona generalmente a través del gestor de arranque (Bootloader) como GRUB (**GR**and **U**nified **B**ootloader también conocido como GNU GRUB).

También se puede lograr mediante la utilización de máquinas virtuales.

- e) ¿Dónde se encuentra ubicado dentro del FileSystem?

La finalidad del Bootloader es la de cargar una imagen de Kernel (SO) de alguna partición para su ejecución. Se ejecuta luego del código de la BIOS.

Existen 2 modos de instalación:

- En el MBR (puede llegar a utilizar MBR Gap)
- En el sector de arranque de la partición o raíz activa (Volume Boot Record)

El MBR es un sector reservado del disco físico (Cilindro 0, Cabeza 0, Sector 1).

En el MBR solo se encuentra la fase 1 del que solo se encarga de cargar la fase 1.5.

La fase 1.5 se ubica en los siguientes 30KB del disco (MBR Gap) y carga la fase 2.

La fase 2 es la interfaz de usuario y carga el Kernel seleccionado.

Se configura a través del archivo /boot.

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

El Kernel es monolítico (un solo proceso) híbrido: Los drivers y código del Kernel



se ejecutan en modo privilegiado con acceso irrestricto al Hardware, algunos se ejecutan en espacio de usuario. Lo que lo hace híbrido es la capacidad de cargar y descargar funcionalidad a través de módulos.

#### 5) Intérprete de comandos (Shell):

##### a) ¿Qué es?

Es la forma que tiene el usuario de comunicarse con el Sistema Operativo. Recibe comandos que transforma en instrucciones de bajo nivel para poder ejecutar.

##### b) ¿Cuáles son sus funciones?

Permite al usuario comunicarse mediante comandos de texto con el sistema operativo, el intérprete es el encargado de transformar los comandos en algo que el Sistema Operativo puede ejecutar.

##### c) Menciona al menos 3 intérpretes de comandos que posee GNU/Linux y compárelos entre ellos:

Bourne Shell (sh), Korn Shell (ksh), Bourne Again Shell (bash)

- Bourne Shell: Sus características incluyen la **simplicidad** debido a su enfoque en la ejecución de comandos simples y scripts, **portabilidad** gracias a que 'sh' está disponible en casi todos los sistemas Unix, su **sintaxis básica**, que soporta operaciones básicas de control de flujo como 'if', 'for', 'while', pero carece de características avanzadas como la edición en la línea de comandos y permite la **utilización de variables** de enterno, además de operaciones simples con ellas.
- Korn Shell: Cuenta con **compatibilidad** con 'sh', por lo tanto todos los scripts creados en 'ksh' pueden ser ejecutados en Bourne Shell sin problema, además posee **funciones avanzadas** como la edición de comandos (similar a 'vi' y 'emacs'), alias y funciones de autocompletado, permite la utilización de **arrays**, facilitando la manipulación de conjunto de datos, una **programación robusta** y es conocido por su buen **rendimiento**.
- Bourne Again Shell: Fue desarrollado con la idea de ser el reemplazo libre de 'sh', pero con más características. **Compatibilidad** total con 'sh' y en gran parte con 'ksh', **interactividad** para la edición en la línea de comandos, autocompletado, historial de comandos, alias y scripts más complejos. También cuenta con **funciones**

**avanzadas** como programación estructurada con funciones, arrays y operaciones aritméticas avanzadas, **scripting poderoso** gracias a las mejoras para la redirección de E/S, procesamiento de señales y manipulación de variables y una gran **popularidad**, siendo el shell predeterminado en la mayoría de distribuciones de Linux, haciéndolo uno de los más usados.

Característica	Bourne Shell (sh)	Korn Shell (ksh)	Bourne Again Shell (bash)
Año de creación	1979	1980s	1989
Compatibilidad con sh	N/A	Completa	Completa
Edición de línea de comandos	No	Sí, con vi/emacs	Sí, con vi/emacs
Autocompletado	No	Sí	Sí
Historial de comandos	No	Sí	Sí
Soporte para arrays	No	Sí	Sí
Alias de comandos	No	Sí	Sí
Popularidad	Alta en sistemas Unix	Alta en sistemas empresariales	Altísima en Linux

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

Los comandos externos al Shell se ubican en /usr/bin, directorio donde están los archivos binarios y ejecutables.

```
leo@leo:~/Documents/GitHub/Facultad/Segundo/Segundo semestre/ISO/Prácticas$ type mkdir
mkdir is /usr/bin/mkdir
```

Los comandos propios no se encuentran en ningún directorio, están integrados directamente en el propio intérprete Shell.

```
leo@leo:~/Documents/GitHub/Facultad/Segundo/Segundo semestre/ISO/Prácticas$ type cd
cd is a shell builtin
```

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

El Shell no es parte del Kernel, es la forma que tenemos de comunicarnos con tal. Cada uno puede utilizar una versión distinta, propia o customizada de Shell debido a que no es algo integrado al Sistema Operativo.

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

El shell predeterminado de un usuario es el programa que se inicia cuando ese usuario inicia sesión en el sistema.

Los Shells más comunes en Linux incluyen:

- Bash 'bin/bash'
- Zsh 'bin/zsh'
- Ksh 'bin/ksh'

Entre otros.

Con el comando 'getent passwd usuario' se puede identificar el Shell predeterminado para el usuario.

Utilizando 'sudo chsh -s /ruta/nuevoShell usuario' se puede cambiar el Shell predeterminado.

```
leo@leo:~/Documents/GitHub/Facultad/Segundo/Semestre/ISO/Prácticas$ getent passwd leo
leo:x:1000:1000:Leo:/home/leo:/bin/bash
```

Para ver el listado de los Shells disponibles se puede utilizar:

```
leo@leo:~$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/usr/bin/sh
/bin/bash
/usr/bin/bash
/bin/rbash
/usr/bin/rbash
/usr/bin/dash
```

## 6) Sistema de Archivos (FileSystem):

### a) ¿Qué es?

El FileSystem o Sistema de Archivos es la estructura lógica que un Sistema Operativo usa para organizar y gestionar los datos en un dispositivo de almacenamiento.

**Estructura del Sistema de Archivos:** En los sistemas Unix y Linux todo comienza del directorio raíz ('/'). Desde ahí se organiza todo el sistema de archivos en una estructura jerárquica de directorios y subdirectorios.

**Directorios** (carpetas en Windows) son contenedores de archivos y otros directorios.

**Archivos** son contenedores de datos que pueden ser de distintos tipos.

**Tipos comunes del FileSystem: Extended** (ext2, ext3, etc): Son sistemas de archivos nativos de Linux.

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

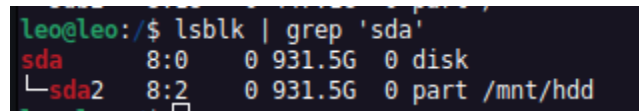
**FAT32 y exFAT**: Sistemas de archivos compatibles con múltiples sistemas operativos, comúnmente en unidades USB.

**HFS+ y APFS**: Sistemas de archivos usados por MacOS.

**Particiones**: División del disco en segmentos independientes, cada uno con su propio sistema de archivos. En Linux, las particiones se montan en puntos de montaje dentro de la estructura del sistema de archivos.

**Montajes** ('mount'): Es el proceso de hacer accesible una partición o un dispositivo en un punto específico del sistema de archivos. Por ejemplo, un disco duro externo puede montarse en '/media/usb'.

Por ejemplo, en mi caso tengo el disco HDD montado en /mnt/hdd.



```
leo@leo:/$ lsblk | grep 'sda'
sda      8:0    0 931.5G  0 disk
└─sda2   8:2    0 931.5G  0 part /mnt/hdd
```

Comando usado: lsblk: Lista la información sobre dispositivos de bloque (almacenamiento), con el símbolo pipe ('|') se pasa el resultado del comando al siguiente, **grep 'sda'**, filtra la salida del comando con la palabra 'sda'.

- b) Mencione sistemas de archivos soportados por GNU/Linux.

Extended (ext3, ext4, etc), NTFS, FAT32, exFAT, etc.

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

Para poder ver el contenido de un dispositivo es necesario realizar un montaje, de esta forma se podrán realizar lecturas y escrituras sobre el contenido de la partición o disco independientemente del formato en el que esté, siempre y cuando sea soportado por GNU/Linux.

- d) ¿Cuál es la estructura básica de los FileSystem 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?

Estructura del Sistema de Archivos: En los sistemas Unix y Linux todo comienza del directorio raíz ('/'). Desde ahí se organiza todo el sistema de archivos en una estructura jerárquica de directorios y subdirectorios.

Se puede imaginar como un árbol invertido, donde la raíz está arriba y se desprenden hacia abajo archivos y directorios.

FHS = FileSystem Hierarchy Standard

Los directorios más importantes según el FHS son:

- / Tope de la estructura de directorios (Similar al C:\ en Windows).
- /home Se almacenan archivos de usuarios (Mis Documentos).
- /var Información que varía de tamaño (Logs, DB, spools).
- /etc Archivos de configuración.
- /bin Archivos binarios y ejecutables.
- /dev Enlace a dispositivos.
- /usr Aplicaciones de usuarios.
- /mnt Subdirectorios donde se montan otras particiones del disco, etc.
- /temp Ficheros temporales.

## 7) Particiones:

### a) Definición. Tipos de particiones. Ventajas y desventajas.

Una partición del disco es una división lógica del espacio total disponible. Usualmente son tratadas como un disco físico independiente y pueden tener distintos tipos de FileSystem (FS) distintos.

**Ventajas:** Permite separar el Sistema Operativo de los archivos del usuario. Se puede crear una partición de Restore de todo el sistema. Poder ubicar el Kernel en una partición de solo lectura.

**Desventajas:** Se reduce el tamaño real del disco, creando la posibilidad de que no entren archivos muy grandes y sea necesario extender (deshacer) particiones.

**Partición primaria:** División cruda del disco. Se almacena información de la misma en el MBR.

**Partición extendida:** Sirve para contener unidades lógicas en su interior. Solo puede existir una partición de este tipo por disco. No se define un tipo de FS directamente sobre ella.

**Partición lógica:** Ocupa la totalidad o parte de la partición extendida y se le define un tipo de FS (partición dentro de partición). Las particiones de este tipo se conectan como una lista enlazada.

Debido al tamaño acotado del MBR para la tabla de particiones:

- Se restringe a 4 la cantidad de particiones primarias.
- 3 primarias y una extendida con sus respectivas particiones lógicas.

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

SCSI: /dev/sd@# (Small Computer System Interface - HDD, CD/DVD, etc).  
SATA: /dev/sd@# (Serial Advanced Technology Attachment - HDD, SSD, etc).  
IDE: /dev/hd# (Integrated Drive Electronics - CD, Unidad de Cinta, etc).

@ = Letra, indica el orden en el que se cargan los discos

# = Número de partición

Por ejemplo, un disco SSD bajo el nombre sda con 2 particiones, resulta en:

- sda
  - sda1
  - Sda2

- c) ¿CUántas particiones son necesarias como mínimo para instalar GNU/Linux?  
Nómbrelas indicando tipo de partición, identificación, tipo de FileSystem y punto de montaje.

No hay una cantidad mínima para instalar GNU/Linux, se puede hacer todo en el disco sin particionar. Es recomendable realizar particiones para separar el Sistema Operativo, de los datos de usuario y la memoria SWAP, más no necesario.

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

\*Robado de <https://github.com/agusrnfr>\*

- Algunos sistemas de archivos (p.e. versiones antiguas de sistemas FAT de Microsoft) tienen tamaños máximos más pequeños que los que el tamaño que proporciona un disco, siendo necesaria una partición de tamaño pequeño, para que sea posible el adecuado funcionamiento de este antiguo sistema de archivos.
- Se puede guardar una copia de seguridad de los datos del usuario en otra partición del mismo disco, para evitar la pérdida de información importante. Esto es similar a un RAID, excepto en que está en el mismo disco.
- En algunos sistemas operativos aconsejan más de una partición para funcionar, como, por ejemplo, la partición de intercambio (swap) en los sistemas operativos basados en Linux.
- A menudo, dos sistemas operativos no pueden coexistir en la misma partición, o usar diferentes formatos de disco "nativo". La unidad se particiona para diferentes sistemas operativos.
- Uno de los principales usos que se le suele dar a las particiones (principalmente a la extendida) es la de almacenar toda la información del usuario (entiéndase música, fotos, vídeos, documentos), para que al momento

de reinstalar algún sistema operativo se formatee únicamente la unidad que lo contiene sin perder el resto de la información del usuario

e) ¿Qué tipo de Software para particionar existe? Menciónelos y compare.

\*Robado de <https://github.com/agusrnfr>\*

**Destructivos:** permiten crear y eliminar particiones (fdisk).

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

8) Arranque (bootstrap) de un Sistema Operativo:

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

El BIOS, Basic I/O System (Sistema Básico de Entrada/Salida) tiene como propósito fundamental iniciar y probar el hardware del sistema y cargar un gestor de arranque o un sistema operativo desde un dispositivo de almacenamiento de datos. Además, provee una capa de abstracción para el hardware, por ejemplo, que consiste en una vía para que los programas de aplicaciones y los sistemas operativos interactúen con el teclado, el monitor y otros dispositivos de I/O. Las variaciones que ocurren en el hardware del sistema quedan ocultas por el BIOS, ya que los programas usan servicios de BIOS en lugar de acceder directamente al hardware. Los sistemas operativos modernos ignoran la capa de abstracción provista por el BIOS y acceden al hardware directamente.

- Está grabado en un chip (ROM, NVRAM).
- En otras arquitecturas también existe, pero se lo conoce con otro nombre
  - Power on Reset + IPL en mainframe.
  - OBP (Open Boot PROM) en SPARC.
- Carga el programa de booteo (desde el MBR).
- El gestor de arranque lanzado desde el MBC carga el Kernel
  - Prueba y hace disponibles los dispositivos.
  - Luego pasa el control al proceso init.
- El proceso de arranque se ve como una serie de pequeños programas de ejecución encadenada.
- La última acción del BIOS es leer el MBC del MBR.

MBC = Master Boot Code

MBR = Master Boot Record

b) ¿Qué es el UEFI? ¿Cuál es su función?

La UEFI, Unified Extensible Firmware Interface (Interfaz Unificada de Firmware Extensible) es una especificación que define una interfaz entre el Sistema Operativo y el Firmware. UEFI reemplaza a la “antigua” interfaz BIOS.

Emplea dos tipos de servicios:

Servicios de arranque: Incluyen funcionalidad requerida por un gestor de arranque o durante el inicio de un Kernel, como texto e interfaz gráfica orientado a una consola, gestión de dispositivos, buses, dispositivos de bloque y sistemas de archivos.

Servicios de ejecución: Incluyen la fecha y la hora, NVRAM y las variables EFI.

c) ¿Qué es el MBR? ¿Cuál es su función?

El MBR, Master Boot Record (Registro de Arranque Principal/Maestro) es el **primer sector de un dispositivo de almacenamiento de datos**, como un disco duro. A veces, se emplea para el arranque del sistema operativo con bootstrap, otras veces es usado para almacenar una tabla de particiones y, en ocasiones, se usa sólo para identificar un dispositivo de disco individual, aunque en algunas máquinas esto último no se usa y es ignorado. Actualmente los discos duros DOS se cuenta con cierta cantidad de particiones, pudiendo ser 3 primarias y 1 extendida, o un máximo de 4 primarias, de las cuales solo se posee una partición “boot” de la cual se arranca todo el sistema operativo.

Si existiese más de un disco rígido en la máquina, sólo uno es designado como “Primary Master Disk”

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

- Los primeros bytes corresponden al MBC, Master Boot Record.
- A partir del byte 446 está la tabla de particiones. Es de 64 bytes.
- Al final existen 2 bytes libres o para firmar el MBR.

El MBC es un pequeño código que permite arrancar el Sistema Operativo.

Si se tiene un SO instalado > Bootloader de MBC típico.

Caso contrario > Bootloader diferente (multietapa).

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

GPT, GUID Partition Table (Tabla de Particiones GUID) es un estándar para la colocación de la tabla de particiones en un disco duro físico. Es parte del estándar EFI (Extensible Firmware Interface) propuesto por Intel para reemplazar el BIOS. La GPT sustituye al MBR usado con el BIOS.



Se utiliza el sistema GPT para solucionar limitaciones del MBR, como la cantidad de particiones.

GPT especifica la ubicación y formato de la tabla de particiones en un disco duro.

Si bien sustituye al MBR, este también se mantiene para asegurar compatibilidad con el esquema BIOS.

GPT usa un modo de direccionamiento lógico (**LBA**, Logical Block Addressing) en lugar de Cylinder-Header-Sector.

El MBR “heredado” 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 (redundancia).

- 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 BIOS.

Para sistemas BIOS es prácticamente obligatorio su uso debido al pequeño tamaño disponible en la MBR para lanzar directamente el Kernel. Con Firmware UEFI se podría lanzar directamente el Kernel pero se usan debido a la versatilidad de opciones que ofrece la carga indirecta a través del bootloader.

Los bootloaders pueden estar repartidos en varias etapas que se ejecutan una tras otra. De ser así, se denominan bootloaders multietapa (Multi Stage Bootloader). Caso contrario, son llamados bootloaders de una sola etapa (Single Stage Bootloader).

Los cargadores de arranque en sistemas BIOS tienen casi siempre al menos dos etapas debido a que el espacio disponible para el cargador de arranque en el MBR es demasiado pequeño. Por tanto casi siempre es necesario saltar a una segunda etapa del cargador de arranque. La ubicación de la segunda etapa dependería del tipo de tabla de particiones usado.

Tabla de particiones tipo DOS ubicadas en el MBR. Es habitual ubicarlo en el espacio entre el MBR y donde empieza la primera partición. Por ejemplo, GRUB con BIOS divide su ejecución en 3 etapas:

- La del MBR
- La del espacio entre el MBR y el inicio de la primera partición
- Dentro del directorio /boot/grub del Linux que haya instalado GRUB

Tabla de particiones GPT. Se ubica en la llamada BIOS Boot Partition que suele estar ubicada en el espacio sobrante entre el final de la GPT y la primera partición.

Existen 2 modos de instalación:

- En el MBR (puede llegar a utilizar MBR Gap).
- En el sector de arranque de la partición raíz o activa (Volume Boot Record).

Bootloaders conocidos: GRUB, LILO, NTLDR, GAG, YaST, etc.

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

- Es el proceso de inicio de una máquina y carga del sistema operativo y se denomina *bootstrap*
- En las arquitecturas x86, el **BIOS (Basic I/O System)** es el responsable de iniciar la carga del SO a través del MBC:
  - Está grabado en un chip (*ROM, NVRAM*)
  - En otras arquitecturas también existe, pero se lo conoce con otro nombre:
    - Power on Reset + IPL en *mainframe*
    - OBP (OpenBoot PROM): en *SPARC*
- Carga el programa de booteo (desde el MBR)
- El gestor de arranque lanzado desde el MBC carga el Kernel:
  - Prueba y hace disponibles los dispositivos
  - Luego pasa el control al proceso **init**
- El proceso de arranque se ve como una serie de pequeños programas de ejecución encadenada

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

\*Robado de <https://github.com/agusrnfr>\*

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 llamado MBR que se carga en memoria y se ejecuta el MBC. Este puede ser de varios tipos, en el caso de Linux, el más frecuentemente usado era LILO, pero ya hace tiempo que se usa en bastantes distribuciones un cargador alternativo, llamado GRUB. Otros Sistemas Operativos tienen su propio programa cargador. Usaremos LILO en la descripción, pues es más ilustrativo.

En el caso concreto de LILO, lo que se carga en el sector de arranque es una parte de éste, denominada "first stage boot loader" (primer paso del cargador de inicio). Su misión es cargar y ejecutar el segundo paso del cargador de inicio. Esta segunda parte suele mostrar una selección de Sistemas Operativos a cargar, procediendo a cargar a continuación el sistema escogido por el usuario (o bien el que se haya predeterminado como sistema por defecto, tras un tiempo de espera, si no escogemos nada). Esta información está incluida dentro del cargador de inicio y, para introducirla, se usa la orden 'lilo' que a su vez usa el contenido de '/etc/lilo.conf'. Todo ello sucede, por supuesto, con el ordenador ya en marcha.

Una vez LILO ha cargado el "kernel" (núcleo) de Linux, le pasa el control a éste. Al cargarlo, le ha pasado algunos parámetros. De éstos, el más importante es el que le dice al núcleo qué dispositivo usar como sistema de ficheros raíz, es decir, lo que en UNIX se denomina '/'. En un ordenador de sobremesa, la raíz sería típicamente una partición de un disco duro, pero en sistemas incrustados es frecuente usar como raíz una partición virtual basada en memoria (Flash, RAM,...). Si el núcleo ha conseguido montar el sistema de ficheros raíz, lo siguiente a ejecutar es el programa 'init'. Sólo si dicho programa es estático (es decir, no usa librerías de funciones externas), no será necesario tener acceso a dichas librerías en la raíz. La librería básica en todo sistema GNU/Linux es la librería estándar C, "glibc". En un sistema mínimo, es decir, con una funcionalidad muy concreta, inmutable y sencilla, con tener solamente el programa 'init' enlazado estáticamente sería suficiente (y el núcleo, claro). En ese caso, init sería en realidad nuestro programa de aplicación al completo. En general, 'init' es sólo el programa que se encarga de arrancar el resto de procesos que la máquina debe ejecutar. Entre sus tareas está el comprobar y montar sistemas de archivos, así como iniciar programas servidores (daemons) para cada función necesaria. Otra tarea importante es la de arrancar procesos 'getty' cuya misión es proporcionar consolas donde poder registrarse y entrar en el sistema. Las órdenes a seguir por 'init' están en el fichero '/etc/inittab'. A partir de ese punto, y en función del sistema de inicialización utilizado (el más frecuente es el denominado "System V") el proceso seguido por 'init' es distinto, pero en el fondo obedece más a un factor de forma, es decir, a una estrategia de ordenamiento de los "scripts" de inicialización de los distintos procesos que a un factor de fondo. Una vez iniciados todos los servidores y procesos de entrada de usuario, o bien estamos delante de una consola de texto en la que el ordenador nos pide que nos identifiquemos, o bien estamos ante una consola gráfica que nos pide lo mismo, o bien estamos ante una pantalla llena de opciones sobre qué ejecutar (escuchar música, ver películas, por ejemplo) si el sistema arranca bajo un usuario predeterminado y no nos pide registrarnos. Esto es, si es que hablamos de un ordenador de sobremesa que, típicamente, nos ofrece una interfaz basada en dispositivos de entrada (teclado, ratón, mando a distancia) y de salida (monitor, TV, audio) para interactuar con él. Pero si el ordenador que se ha iniciado es un dispositivo con una funcionalidad concreta y su misión es

controlar una serie de procesos y accedemos a él a través de medios indirectos (como pueda ser un navegador Web), el ordenador se inicia cuando está en disposición de prestar sus servicios, aun cuando no haya una indicación visual de dicho estado.

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

\*Robado de <https://github.com/agusrnfr>\*

El comando para finalizar correctamente un sistema Linux es shutdown. Se utiliza generalmente de una de dos maneras diferentes:

- Si Ud. es el único usuario del sistema, debe finalizar todos los programas que estén en ejecución, finalizar todas las sesiones (log out) de todas las consolas virtuales, e iniciar una sesión como usuario root (o mantener la sesión si ya existe una, pero debe cambiar de directorio de trabajo al directorio HOME de root, para evitar problemas al desmontarse los sistemas de archivos).

Finalmente ejecute el comando shutdown -h now. Si desea postergar durante algún lapso el comando shutdown, reemplace now con un signo + (mas) y un número que indica minutos de espera.

- Alternativamente, si el sistema está siendo utilizado por muchos usuarios, utilice el comando shutdown -h +time mensaje, donde time es el número de minutos en que se posterga la detención del sistema, y el mensaje es una explicación breve del porqué se está apagando el sistema. # shutdown -h +10 'We will install a new disk. System should > be back on-line in three hours.' # El ejemplo advierte a todos los usuarios que el sistema se apagará en diez minutos, y que sería mejor que se desconectaran o se arriesgan a perder la información con la que están trabajando. La advertencia se muestra en cada terminal donde existe un usuario conectado, incluyendo las xterm (emuladores de terminales para el sistema X Window).

- i) ¿Es posible tener en una PC GNU/Linux y otro Sistema Operativo instalado? Justifique su respuesta.

Es posible tener dos Sistemas Operativos instalados en simultáneo pero se pueden usar de a uno. También se puede armar un dual boot y acceder a cualquiera mediante un gestor de arranque.

La única limitación es el espacio disponible en el disco o las particiones correspondientes a la instalación de cada Sistema Operativo.

## 9) Archivos:

- a) ¿Cómo se identifican los archivos en GNU/Linux?
- b) Investigue el funcionamiento de los editores “vi” y “mcedit”, y los comandos “cat” y “more”.

- c) Cree un archivo llamado “prueba.exe” en su directorio personal usando el “vi”. El mismo debe contener su número de alumno y nombre.
- d) Investigue el funcionamiento del comando file. Pruébelo con diferentes archivos.  
¿Qué diferencia nota?

El comando “file” indica el tipo del archivo que se pasa como parámetro.

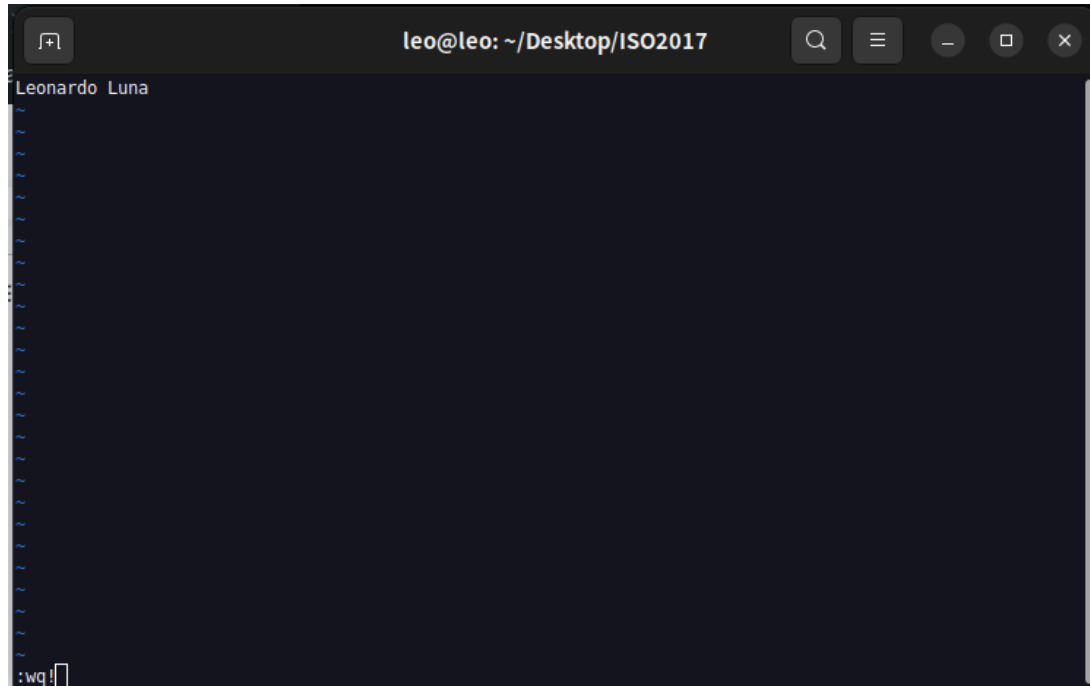
10) Indique qué commando es necesario utilizar para realizar cada una de las siguientes acciones:

- a) Cree la carpeta ISO2017.
- b) Acceda a la carpeta (cd).
- c) Cree dos archivos con los nombres “iso2017-1” e “iso2017-2” (touch).
- d) Liste el contenido del directorio actual (ls).
- e) Visualizar la ruta donde estoy situado (pwd).
- f) Busque todos los archivos en los que su nombre contiene la cadena “iso\*” (find).
- g) Informe la cantidad de espacio libre en disco (df).
- h) Verifique los usuarios conectados al sistema (who).
- i) Acceder al archivo “iso2017-1” e ingresar Nombre y Apellido.
- j) Mostrar en pantallas las últimas líneas de un archivo (tail).

```

leo@leo: ~/Desktop/ISO2017
leo@leo:~/Desktop$ mkdir ISO2017
leo@leo:~/Desktop$ cd ISO2017/
leo@leo:~/Desktop/ISO2017$ touch iso2017-1
leo@leo:~/Desktop/ISO2017$ touch iso2017-2
leo@leo:~/Desktop/ISO2017$ ls
iso2017-1  iso2017-2
leo@leo:~/Desktop/ISO2017$ pwd
/home/leo/Desktop/ISO2017
leo@leo:~/Desktop/ISO2017$ find . -name "iso*"
./iso2017-2
./iso2017-1
leo@leo:~/Desktop/ISO2017$ df
Filesystem      1K-blocks      Used Available Use% Mounted on
tmpfs           1571788        2384   1569404    1% /run
/dev/sdb2       460364840 113301784 323604292   26% /
tmpfs           7858932       272820   7586112    4% /dev/shm
tmpfs           5120          20       5100      1% /run/lock
tmpfs           7858932          0   7858932    0% /run/qemu
tmpfs           1571784        228   1571556    1% /run/user/1000
leo@leo:~/Desktop/ISO2017$ who
leo    seat0      2024-08-30 15:23 (login screen)
leo    :0         2024-08-30 15:23 (:0)
leo@leo:~/Desktop/ISO2017$ tail iso2017-1
Leonardo Luna
leo@leo:~/Desktop/ISO2017$

```

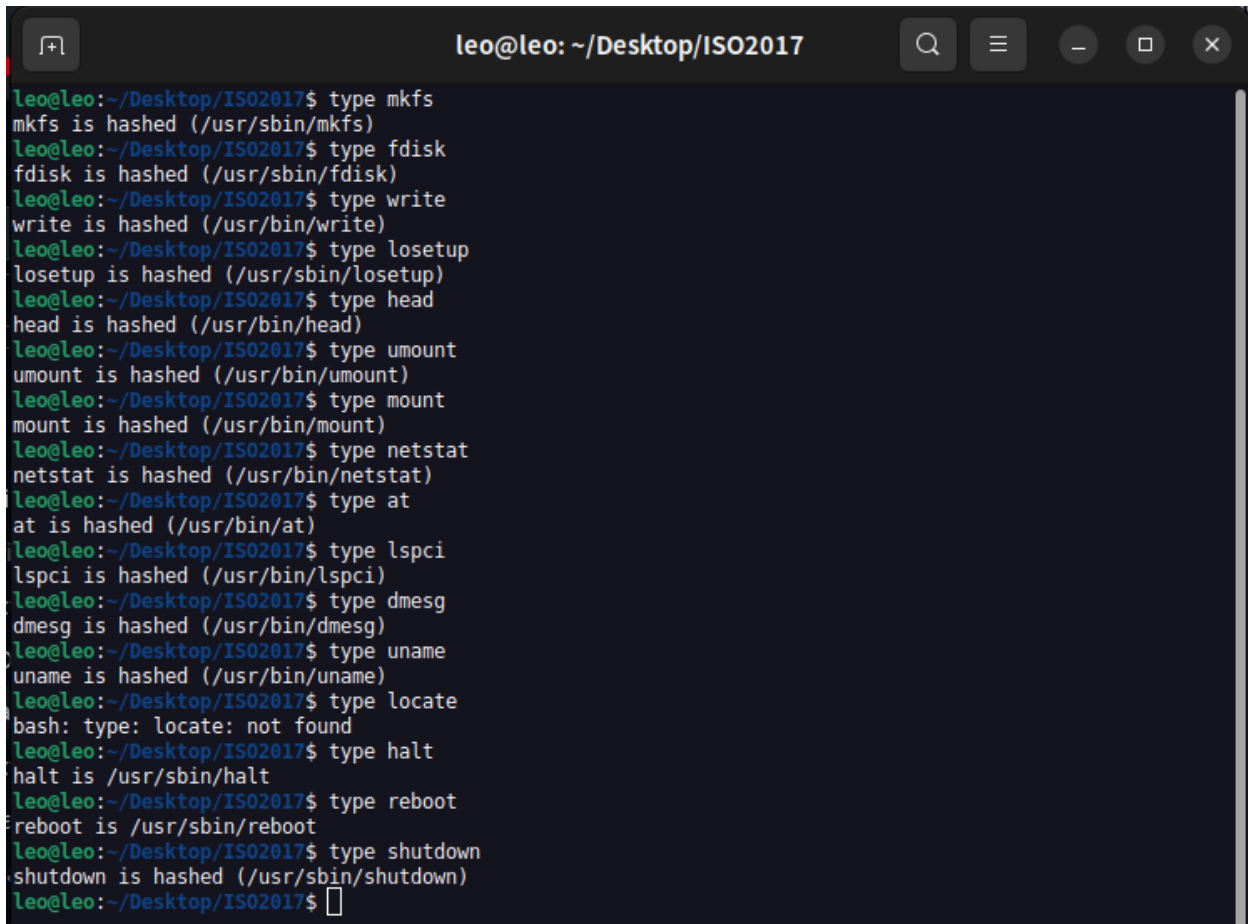


11) Investigue su funcionamiento y parámetros más importantes:

- a) Shutdown: Apagar, reiniciar o detener el sistema.  
shutdown [OPTIONS] [TIME] [MESSAGE]  
Opciones: -r, -h, -H, -P, -c, -k
- b) Reboot: Reiniciar el sistema de manera controlada.  
reboot [OPTIONS]  
Opciones: -halt, -p, -reboot, -f, -w
- c) Halt: Para el procesador (No es un reinicio).  
halt [OPTIONS]  
Opciones: -l, -n, -p, -q, -y
- d) Locate: Encontrar archivos por su nombre.  
locate [OPTIONS] [FILENAME]  
Opciones: -i, -l, -b, -S, -e, -L, -P, -A, -r, -q, -n
- e) Uname: Mostrar información del sistema.  
uname [OPTIONS]  
Opciones: -a, -s, -n, -v, -m, -p, -i, -o
- f) Dmesg: Muestra y controla el Kernel Ring Buffer.  
dmesg [OPTIONS]  
Opciones: -C, -c, -D, -E, -F, -f, -H, -J, -k, -L, -l, -n, -P, -p, -r, -S, -s, etc.

- g) Lspci: Listar los buses PCI y dispositivos del sistema.  
lspci [OPTIONS]  
Opciones: -v, -nn, -k, -t, -i, -d, -s, -x, -xxx, -A, -D, -F
- h) At: Permite programar tareas o acciones recurrentes.  
at [OPTIONS]  
Opciones: -m, -f, -l, -d, -v, -q, -c
- i) Netstat: Muestra información y diagnosis relacionados a la red.  
netstat [OPTIONS]  
Opciones: -at, -au, -l, -s, -tp
- j) Mount: Montar un FileSystem a un dispositivo.  
mount [OPTIONS] [SOURCE] [DIRECTORY]  
Opciones: -a, -c, -f, -F, -T, -i, -l, -m, -n
- k) Umount: Desmontar un FileSystem de un dispositivo.  
umount [OPTIONS] [SOURCE] | [DIRECTORY]  
Opciones: -a, -A, -c, -d, -f, -i, -n, -l, -O, -R, -r, -t, -v, -q, -N, -h, -V
- l) Head: Imprimir las primeras 10 líneas de un archivo.  
head [OPTIONS] [FILE]  
Opciones: -c, -n, -q, -v, -z
- m) Losetup: Crear y controlar dispositivos de loop.  
losetup [OPTIONS] [LOOPDEV] [FILE]  
Opciones: -a, -d, -D, -f, -c, -j, -L, -o, -b, -P, -r, -v, -J, -l, -n, -O, -h, -v
- n) Write: Enviar un mensaje a otro usuario.  
write [OPTIONS] [USER] [TTYNAME]  
Opciones: -h, -v
- TTY = **TeleTY**pewriter
- o) Mkfs: "Make File System", crear un FileSystem de Linux.  
mkfs [OPTIONS] [-t TYPE] [FS-OPTIONS] [DEVICE] [SIZE]  
Opciones: -t, -V, -h, --version
- p) Fdisk (con cuidado): Mostrar o manipular una tabla de particiones.  
fdisk [OPTIONS] [DISK]  
Opciones: -b, -B, -c, -L, -l, -x, -n, -o, -t, -u, -s, -w, -W, -C, -H, -S, -h, -V

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



```
leo@leo: ~/Desktop/ISO2017
leo@leo:~/Desktop/ISO2017$ type mkfs
mkfs is hashed (/usr/sbin/mkfs)
leo@leo:~/Desktop/ISO2017$ type fdisk
fdisk is hashed (/usr/sbin/fdisk)
leo@leo:~/Desktop/ISO2017$ type write
write is hashed (/usr/bin/write)
leo@leo:~/Desktop/ISO2017$ type losetup
losetup is hashed (/usr/sbin/losetup)
leo@leo:~/Desktop/ISO2017$ type head
head is hashed (/usr/bin/head)
leo@leo:~/Desktop/ISO2017$ type umount
umount is hashed (/usr/bin/umount)
leo@leo:~/Desktop/ISO2017$ type mount
mount is hashed (/usr/bin/mount)
leo@leo:~/Desktop/ISO2017$ type netstat
netstat is hashed (/usr/bin/netstat)
leo@leo:~/Desktop/ISO2017$ type at
at is hashed (/usr/bin/at)
leo@leo:~/Desktop/ISO2017$ type lspci
lspci is hashed (/usr/bin/lspci)
leo@leo:~/Desktop/ISO2017$ type dmesg
dmesg is hashed (/usr/bin/dmesg)
leo@leo:~/Desktop/ISO2017$ type uname
uname is hashed (/usr/bin/uname)
leo@leo:~/Desktop/ISO2017$ type locate
bash: type: locate: not found
leo@leo:~/Desktop/ISO2017$ type halt
halt is /usr/sbin/halt
leo@leo:~/Desktop/ISO2017$ type reboot
reboot is /usr/sbin/reboot
leo@leo:~/Desktop/ISO2017$ type shutdown
shutdown is hashed (/usr/sbin/shutdown)
leo@leo:~/Desktop/ISO2017$
```