Introducción a los Sistemas Operativos

# Práctica 1

1. Características de *GNU/Linux*:

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

Características Generales:

• Es multiusuario. Un sistema operativo multiusuario es aquel que es capaz de proveer servicio y procesamiento a múltiples usuarios al mismo tiempo.

• Es multitarea y multiprocesador. Esto significa que puede realizar varias tareas al mismo tiempo y que puede utilizar varios procesadores para realizar estas tareas.

• Es altamente portable

• Posee diversos intérpretes de 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 (/temp, /home, etc.)

• Es case sensitive

• Es codigo abierto

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

Windows / Linux

* Uso estándar: Interfaz gráfica de usuario / Líneas de comandos
* Acceso remoto: Servidor de terminales; el cliente tiene que instalarse y configurarse / Solución integrada (terminal y shell)
* Software y características: Soporta programas habituales; posibilidad de utilizar aplicaciones de Microsoft / No ofrece portabilidad para todos los programas; gran cantidad de aplicaciones disponibles
* Soporte de hardware: El nuevo hardware está diseñado normalmente para los sistemas Windows / Por lo general, pueden utilizarse los controladores de hardware para las distribuciones de Linux más tarde
* Seguridad : Elevado potencial de errores de usuario; interfaz integrada como posible punto de ataque / Los usuarios habituales no tienen acceso a los ajustes básicos del sistema; las vulnerabilidades conocidas se solucionan rápidamente
* Asistencia : Asistencia a largo plazo para todas las versiones / La asistencia varía en función de la distribución y de la versión
* Documentación El sistema y sus aplicaciones están muy bien documentadas, algo que difiere de los componentes de la API y de los formatos de los datos // Se conoce el código fuente completo del sistema, las API, las bibliotecas y las aplicaciones; la mayoría de manuales y de páginas informativas están en inglés

(c) ¿Qué es **GNU**?

GNU: Es un Sistema Operativo tipo Unix (Unix like), pero libre, S.O. diseñado por miles de programadores, S.O. gratuito y de libre distribución (se baja desde la Web, CD, etc.). Existen diversas distribuciones (customizaciones). Es de código abierto, lo que nos permite estudiarlo, personalizarlo, auditarlo, aprovecharnos de la documentación, etc...

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

GNU = GNU No es Unix

Iniciado por Richard Stallman en 1983 con el fin de crear un Unix libre (el sistema GNU).Para asegurar que el mismo fuera libre, se necesito crear un marco regulatorio conocido como GPL (General Public License de GNU).En 1985, Stallman crea la FSF (Free Software Foundation), con el fin de financiar el proyecto GNU.En 1990, GNU ya contaba con un editor de textos (Emacs), un compilador (GCC) y gran cantidad de bibliotecas que componen un Unix tıpico.Faltaba el componente principal → El núcleo (Kernel) GNU. Si bien ya se venía trabajando en un núcleo conocido como TRIX, es en 1988 que se decide abandonarlo debido a su complejidad (corría en hardware muy costoso). En este momento se decide adoptar como base el núcleo MACH para crear GNU Hurd, el cual tampoco prospero.

Linus Torvalds ya venía trabajando desde 1991 en un Kernel denominado Linux, el cual se distribuiría bajo licencia GPL .En el año 1992, Torvalds y Stallman deciden fusionar ambos proyectos, y es allí donde nace GNU/Linux. GNU/Linux pertenece al desarrollo del software libre.

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

La multitarea es la característica de los sistemas operativos modernos que permite que varios procesos o aplicaciones se ejecuten aparentemente al mismo tiempo, compartiendo uno o más procesadores.

Linux es de multitarea y multiprocesador, porque puede haber varias tareas.

(f) ¿Qué es **POSIX**?

POSIX es un acrónimo que significa "Portable Operating System Interface" (Interfaz Portátil de Sistema Operativo, en español). Es un conjunto de estándares que especifican la interfaz de programación de aplicaciones (API), la utilidad de comandos, la definición de formato de archivos y las interfaces de línea de comandos para sistemas operativos compatibles con POSIX.

El objetivo principal de POSIX es permitir que el software se escriba una vez y se ejecute en cualquier sistema operativo que cumpla con los estándares de POSIX. Esto es particularmente importante para aplicaciones empresariales y de misión crítica que necesitan ser portables y ejecutarse en diferentes sistemas operativos.

POSIX se originó en el sistema operativo Unix y ha sido adoptado por muchos sistemas operativos modernos, incluidos Linux, macOS y sistemas operativos basados en Unix. La adopción de los estándares de POSIX garantiza la compatibilidad entre los sistemas operativos, lo que facilita la interoperabilidad y la portabilidad del software.

2. Distribuciones de *GNU/Linux*

(a) ¿Qué es una distribución de *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 completo que se compone del núcleo de Linux, herramientas y utilidades de GNU, y una amplia variedad de software adicional de código abierto que se empaqueta y se distribuye junto con el sistema operativo. Cada distribución de GNU/Linux está diseñada para satisfacer diferentes necesidades y preferencias de los usuarios, y puede variar en su selección de software preinstalado, la forma en que se manejan las actualizaciones, la facilidad de uso y otros factores.

Ejemplos de distribuciones de GNU/Linux y algunas diferencias básicas entre ellas:

1. Ubuntu: Es una de las distribuciones de GNU/Linux más populares y fáciles de usar. Está diseñado para ser fácil de instalar y usar, y viene preinstalado con una amplia variedad de software de código abierto, incluyendo una suite de oficina completa, un navegador web y un reproductor multimedia. Ubuntu se basa en la distribución de Debian y utiliza el gestor de paquetes apt para la gestión de software.
2. Fedora: Es una distribución de GNU/Linux comunitaria respaldada por Red Hat. Se centra en ofrecer las últimas versiones de software de código abierto, lo que lo convierte en una buena opción para usuarios avanzados y desarrolladores. Fedora utiliza el gestor de paquetes DNF para la gestión de software.
3. Debian: Es una distribución de GNU/Linux estable y madura que es conocida por su fiabilidad y facilidad de uso. Es uno de los sistemas operativos más antiguos y ampliamente utilizados en el mundo de GNU/Linux. Debian utiliza el gestor de paquetes apt para la gestión de software.
4. Arch Linux: Es una distribución de GNU/Linux enfocada en la simplicidad y la elegancia. Está diseñada para usuarios avanzados y ofrece un sistema de paquetes de software flexible y personalizable llamado "pacman". Arch Linux es conocida por su documentación detallada y su comunidad activa y comprometida.

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

Las distribuciones de GNU/Linux pueden diferir en varios aspectos, como la selección de software preinstalado, la facilidad de uso, la estabilidad, la seguridad, la velocidad, la personalización y la documentación. Aquí hay algunas áreas en las que las distribuciones de GNU/Linux pueden diferir:

* Gestor de paquetes: Las distribuciones de GNU/Linux pueden utilizar diferentes gestores de paquetes para la gestión de software. Los gestores de paquetes son programas que permiten a los usuarios instalar, actualizar y desinstalar software en el sistema. Algunas distribuciones utilizan el gestor de paquetes apt, mientras que otras pueden usar DNF, pacman, zypper, entre otros.
* Desktop environment: Las distribuciones de GNU/Linux también pueden utilizar diferentes entornos de escritorio. Los entornos de escritorio son interfaces gráficas de usuario que permiten a los usuarios interactuar con el sistema operativo. Algunos ejemplos de entornos de escritorio populares incluyen GNOME, KDE, Xfce, LXDE y Cinnamon.
* Enfoque: Las distribuciones de GNU/Linux pueden tener diferentes enfoques. Algunas se centran en la estabilidad y la seguridad, mientras que otras priorizan la velocidad y el rendimiento. Algunas distribuciones están diseñadas para usuarios avanzados y desarrolladores, mientras que otras están diseñadas para ser fáciles de usar para los usuarios novatos.
* Comunidad: Las distribuciones de GNU/Linux también pueden tener diferentes comunidades. Algunas distribuciones tienen comunidades grandes y activas que ofrecen soporte y documentación detallada, mientras que otras pueden tener comunidades más pequeñas.

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

Debian es una distribución de GNU/Linux estable y madura que se destaca por su enfoque en la libertad de software y su amplia variedad de paquetes de software disponibles. Debian se basa en la filosofía de que el software debe ser libre y estar disponible para todos, y por lo tanto, todo el software en la distribución de Debian es de código abierto.

Según la página "Acerca de Debian" en el sitio web, los objetivos del proyecto son los siguientes:

1. Ofrecer un sistema operativo gratuito y completo basado en Linux y el software de GNU.
2. Mantener altos estándares de calidad y estabilidad.
3. Ser administrado y desarrollado de forma abierta y transparente.
4. Contribuir al software libre y de código abierto en general.

Debian comenzó en 1993 como un proyecto comunitario liderado por Ian Murdock. La primera versión estable, Debian 1.1, fue lanzada en 1996. Desde entonces, el proyecto ha crecido enormemente y se ha convertido en una de las distribuciones de GNU/Linux más populares y utilizadas en todo el mundo. Actualmente, Debian cuenta con una gran comunidad de desarrolladores y usuarios que trabajan juntos para mantener y mejorar la distribución.

Algunas de las versiones más importantes de Debian son:

* Debian 2.0 "Hamm" (1998): Esta fue la primera versión de Debian en utilizar el gestor de paquetes dpkg.
* Debian 3.0 "Woody" (2002): Esta fue la primera versión de Debian en utilizar el núcleo de Linux 2.2 y la primera en incluir el gestor de paquetes apt-get.
* Debian 4.0 "Etch" (2007): Esta versión introdujo mejoras en la seguridad y la instalación de sistemas encriptados.
* Debian 5.0 "Lenny" (2009): Esta versión presentó importantes mejoras en el instalador de Debian y en la compatibilidad con hardware.
* Debian 6.0 "Squeeze" (2011): Esta versión presentó mejoras significativas en la infraestructura de seguridad y en la compatibilidad con múltiples arquitecturas.
* Debian 7.0 "Wheezy" (2013): Esta versión presentó importantes mejoras en la compatibilidad con sistemas UEFI y en el soporte para varias arquitecturas.
* Debian 8.0 "Jessie" (2015): Esta versión presentó mejoras en la estabilidad y la seguridad, así como en la compatibilidad con sistemas ARM.
* Debian 9.0 "Stretch" (2017): Esta versión presentó importantes mejoras en el soporte de hardware, en la estabilidad y la seguridad.
* Debian 10 "Buster" (2019): Esta versión presentó mejoras significativas en la seguridad, la facilidad de uso y el soporte de hardware.

3. Estructura de *GNU/Linux*:

(a) Nombre cuales son los 3 componentes fundamentales de *GNU/Linux*.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Los tres componentes fundamentales de GNU/Linux son:

1. El núcleo o kernel: es el componente central del sistema operativo que se encarga de administrar los recursos del hardware, como el procesador, la memoria, los dispositivos de entrada y salida, etc. El kernel de Linux fue creado por Linus Torvalds en 1991 y es el componente principal de la mayoría de las distribuciones de GNU/Linux.
2. El sistema de archivos: es la estructura que organiza y administra los archivos y directorios en el disco duro del sistema. En GNU/Linux, se utiliza un sistema de archivos jerárquico llamado Filesystem Hierarchy Standard (FHS) que define la estructura de los directorios y archivos del sistema.
3. El conjunto de herramientas de software de GNU: es un conjunto de programas y herramientas desarrollados por el proyecto GNU que se utilizan en conjunto con el kernel de Linux para crear un sistema operativo completo. Entre estas herramientas se encuentran el compilador GCC, la biblioteca C de GNU (glibc), la utilidad de gestión de paquetes APT, el intérprete de comandos Bash, entre otros.

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

La estructura básica del sistema operativo GNU/Linux se basa en el modelo de sistema de archivos jerárquico, donde todos los archivos y directorios del sistema se organizan en una jerarquía que comienza en el directorio raíz ("/"). La estructura básica incluye los siguientes elementos:

* /bin: Contiene los archivos ejecutables de los comandos esenciales del sistema, como ls, cp, mv, rm, etc. Estos comandos se utilizan con frecuencia y se espera que estén disponibles en todos los sistemas GNU/Linux.
* /sbin: Contiene los archivos ejecutables de los comandos del sistema que sólo pueden ser utilizados por el superusuario (root), como shutdown, mount, umount, etc.
* /usr: Contiene los archivos y programas que se instalan después de la instalación inicial del sistema. Incluye subdirectorios como /usr/bin, que contiene programas de usuario, y /usr/sbin, que contiene programas de sistema adicionales.
* /etc: Contiene los archivos de configuración del sistema. Por ejemplo, /etc/passwd contiene información de los usuarios, /etc/fstab contiene información sobre los sistemas de archivos que se montan al inicio del sistema, etc.
* /var: Contiene los archivos y directorios que cambian con frecuencia durante el uso del sistema, como los archivos de registro de sistema en /var/log y los correos electrónicos en /var/mail.
* /tmp: Contiene archivos temporales que son creados por programas y eliminados automáticamente al reiniciar el sistema.
* /home: Contiene los directorios personales de los usuarios del sistema.
* /root: Es el directorio personal del superusuario (root).
* /dev: Contiene los archivos de dispositivos que representan los dispositivos de hardware del sistema.
* /proc: Contiene información sobre los procesos que se están ejecutando en el sistema.
* /boot: Contiene los archivos necesarios para el arranque del sistema, como el kernel y los archivos de configuración de arranque.
* /lib: Contiene las bibliotecas compartidas necesarias para que los programas funcionen correctamente.

4. *Kernel*:

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

El kernel es el componente central del sistema operativo que se encarga de administrar los recursos del hardware, como el procesador, la memoria, los dispositivos de entrada y salida, etc. En el caso de GNU/Linux, el kernel fue creado por Linus Torvalds en 1991 como un proyecto personal para crear un sistema operativo compatible con UNIX, pero que fuera libre y gratuito.

Historia:

En 1991 Linus Torvalds inicia la programación de un Kernel Linux basado en Minix (clon de Unix desarrollado por Tenembaum en 1987 con el fin de crear un S.O. de uso didáctico) y era un kernel muy básico que sólo podía ejecutar un número limitado de comandos y programas. 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.Desarrollo continuado por miles de programadores al rededor del mundo

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. 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 catapultó a GNU/Linux como un SO estable y robusto. Durante este período 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 ´ultima en su revisión . El 17 de julio de 2011 se lanza la versión 3.01. No agrega mayores cambios. La decisión del cambio son los 20 años del SO y no superar los 40 números de revisión. Totalmente compatible con 2.6. La última versión estable es la 5.13.12 (agosto de 2021).

(b) ¿Cuáles son sus funciones principales?

Las funciones principales del kernel de GNU/Linux son las siguientes:

1. Administración de recursos: El kernel es responsable de administrar los recursos del hardware, como el procesador, la memoria, los dispositivos de entrada y salida, etc. Debe garantizar que estos recursos estén disponibles para las aplicaciones que los necesiten y asignarlos de manera justa y eficiente.
2. Administración de procesos: El kernel es responsable de administrar los procesos o programas que se están ejecutando en el sistema. Debe asegurarse de que los procesos se ejecuten de manera segura y eficiente, y que se asignen los recursos necesarios para cada uno.
3. Administración de memoria: El kernel es responsable de administrar la memoria del sistema y de garantizar que las aplicaciones tengan acceso a la memoria que necesitan para funcionar correctamente.
4. Administración de dispositivos: El kernel es responsable de administrar los dispositivos de entrada y salida del sistema, como discos duros, impresoras, teclados, ratones, etc. Debe garantizar que los dispositivos funcionen correctamente y que los programas puedan interactuar con ellos de manera adecuada.
5. Administración de red: El kernel es responsable de administrar la red del sistema y de garantizar que los programas puedan comunicarse a través de la red de manera segura y eficiente.
6. Seguridad: El kernel es responsable de garantizar la seguridad del sistema y de protegerlo contra ataques malintencionados o fallas de seguridad.

En resumen, el kernel de GNU/Linux es el componente central del sistema operativo que se encarga de administrar los recursos del hardware y de proporcionar una interfaz entre las aplicaciones y el hardware del sistema. Sin él, el sistema operativo no podría funcionar correctamente.

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

La versión actual del kernel de GNU/Linux es la 5.11.16, que fue lanzada el 20 de abril de 2021.

Antes de la versión 2.4, el esquema de versionado del kernel de Linux se basaba en tres números separados por puntos, donde el primer número indicaba la versión principal del kernel, el segundo número indicaba una versión menor y el tercer número indicaba una versión de corrección. Por ejemplo, la versión 2.2.1 indicaba que se trataba de la segunda versión principal del kernel, la segunda versión menor y la primera versión de corrección.

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

A partir de la versión 2.6, se cambió el esquema de versionado para incluir un número de versión de cuatro dígitos, donde el primer número indicaba la versión principal del kernel, el segundo número indicaba una versión menor, el tercer número indicaba una versión de corrección y el cuarto número indicaba una versión de parche. Por ejemplo, la versión 2.6.32.1 indicaba que se trataba de la segunda versión principal del kernel, la sexta versión menor, la trigésima segunda versión de corrección y la primera versión de parche.

Este cambio en el esquema de versionado permitió una mayor granularidad y precisión en la numeración de las versiones del kernel y facilitó la identificación de las versiones que incluían correcciones de seguridad importantes y otras mejoras críticas.

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

Sí, es posible tener más de un kernel de GNU/Linux instalado en la misma máquina. De hecho, esta es una práctica común en muchos sistemas, especialmente en servidores y en sistemas donde se requiere un alto nivel de estabilidad y seguridad.

Eso sí, solo ejecutaríamos uno a la vez, el otro quedaría instalado y almacenado en el disco. También podríamos emular varios kernels a partir de un mismo sistema operativo base y ejecutarlos a la vez, aunque serían kernels virtuales y no estarían funcionando realmente a nivel sistema.

(e) ¿Dónde se encuentra ubicado dentro del File System?

El kernel de GNU/Linux se encuentra ubicado en el sistema de archivos, en la ruta "/boot/vmlinuz" o en alguna variante de esa ruta dependiendo de la distribución de GNU/Linux que se esté utilizando.

El directorio "/boot" es donde se almacenan los archivos relacionados con el proceso de arranque del sistema, incluyendo el kernel y los archivos de configuración del gestor de arranque (como GRUB). El archivo "vmlinuz" (que significa "archivo comprimido del kernel de Linux") es el archivo ejecutable que contiene el kernel de GNU/Linux. Dependiendo de la configuración del sistema, puede haber varios archivos relacionados con diferentes versiones del kernel en el directorio "/boot".

Cabe destacar que aunque la ubicación del kernel de Linux suele ser en la ruta mencionada anteriormente, esto puede variar dependiendo de la distribución y la configuración del sistema en particular.

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

El kernel de GNU/Linux se considera un kernel monolítico híbrido. Aunque es cierto que todas las funciones del sistema operativo se encuentran en una sola imagen ejecutable, el kernel de GNU/Linux también permite la carga dinámica de módulos externos en tiempo de ejecución, lo que permite una mayor flexibilidad y modularidad del sistema.

De hecho, la capacidad de cargar módulos externos en tiempo de ejecución es una de las características distintivas del kernel de GNU/Linux en comparación con otros kernels monolíticos. Esto significa que el kernel de GNU/Linux puede ser extendido o modificado sin necesidad de recompilar o reiniciar el sistema, lo que lo hace más adaptable a diferentes escenarios y necesidades.

5. Intérprete de comandos (Shell):

(a) ¿Qué es?

El intérprete de comandos, también conocido como shell, es un programa informático que proporciona una interfaz de usuario para acceder a los servicios y recursos del sistema operativo. En otras palabras, el shell es la interfaz entre el usuario y el sistema operativo.

El shell de GNU/Linux es un programa que acepta comandos escritos por el usuario y los interpreta para realizar acciones en el sistema operativo, como la creación de archivos, la gestión de procesos, la configuración del sistema, entre otras tareas.

(b) ¿Cuáles son sus funciones?

Las funciones principales del intérprete de comandos, o shell, son las siguientes:

1. Interpretar los comandos: el shell interpreta los comandos escritos por el usuario y los convierte en instrucciones que el sistema operativo puede entender y ejecutar.
2. Proporcionar una interfaz de usuario: el shell proporciona una interfaz de usuario para acceder a los servicios y recursos del sistema operativo, lo que permite a los usuarios interactuar con el sistema de manera eficiente.
3. Gestionar los procesos: el shell puede iniciar, detener y gestionar los procesos en el sistema operativo. También puede permitir la ejecución de varios procesos en segundo plano o en paralelo.
4. Gestionar archivos y directorios: el shell proporciona comandos para crear, modificar, mover y eliminar archivos y directorios en el sistema de archivos.
5. Redirigir la entrada y salida: el shell permite redirigir la entrada y salida de los comandos para que los resultados se envíen a un archivo, impresora u otro destino.
6. Personalizar el entorno del usuario: el shell permite al usuario personalizar su entorno de trabajo mediante la configuración de variables de entorno, alias y scripts personalizados.
7. Facilitar la automatización de tareas: el shell permite a los usuarios automatizar tareas repetitivas mediante la creación de scripts y programas personalizados.

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

Los tres más comunes son:

* Bash: El Bourne-Again Shell (Bash) es el intérprete de comandos más utilizado en GNU/Linux y es el predeterminado en la mayoría de las distribuciones. Bash es una versión mejorada del shell original de Unix, el Bourne shell (sh), y proporciona muchas características adicionales, como autocompletado de comandos, historial de comandos y edición de línea de comando.
* Zsh: Z shell (Zsh) es un intérprete de comandos que es similar a Bash, pero con características adicionales. Zsh tiene un autocompletado más sofisticado, una sintaxis más flexible y un sistema de plugins que permite a los usuarios agregar nuevas características. Zsh también proporciona un marco de trabajo de temas y personalización de línea de comando más avanzado que Bash.
* Fish: Friendly Interactive Shell (Fish) es un intérprete de comandos más moderno que fue diseñado para ser fácil de usar y aprender para los usuarios nuevos en la línea de comandos. Fish tiene autocompletado de comandos basado en texto predictivo, resaltado de sintaxis avanzado y una sintaxis más simple y fácil de recordar. Fish también tiene un conjunto más limitado de características que Bash y Zsh, pero es más fácil de configurar y personalizar.

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

En GNU/Linux, los comandos propios del intérprete de comandos (shell) generalmente están ubicados en una de las siguientes rutas o paths:

* /bin
* /usr/bin
* /usr/local/bin

El path /bin contiene comandos esenciales del sistema que se necesitan para arrancar el sistema operativo. El path /usr/bin contiene comandos que se instalan con paquetes de software adicionales. Y /usr/local/bin contiene comandos que se han instalado manualmente por el usuario o administrador del sistema.

En cuanto a los comandos externos al shell, su ubicación puede variar según el software instalado en el sistema. Por lo general, se instalan en uno de los siguientes paths:

* /usr/bin
* /usr/local/bin
* /opt

Además, algunos programas también pueden tener sus propios directorios de instalación. En general, los comandos externos se buscan en todos los paths listados en la variable de entorno $PATH del usuario actual.

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

El shell no es parte del kernel de GNU/Linux porque es un programa independiente que se ejecuta en el espacio de usuario, en lugar de ejecutarse en el espacio del kernel. El kernel de GNU/Linux es el componente central del sistema operativo, que se encarga de la gestión de los recursos del hardware, el control de acceso a los mismos, y la comunicación entre los diferentes programas y procesos en ejecución.

Por otro lado, el shell es un programa que permite a los usuarios interactuar con el sistema operativo, proporcionando una interfaz de línea de comandos o una interfaz gráfica de usuario (GUI). El shell se ejecuta en el espacio de usuario, lo que significa que no tiene acceso directo a los recursos del hardware. En cambio, interactúa con el kernel a través de las llamadas al sistema, que son interfaces definidas por el kernel para permitir que los programas del espacio de usuario realicen operaciones en el espacio del kernel.

La separación entre el kernel y el shell no está directamente relacionada con la seguridad, aunque es cierto que la arquitectura del kernel monolítico puede presentar algunos riesgos de seguridad al tener un gran número de privilegios y funcionalidades incluidas en el mismo componente.

La separación entre el kernel y el shell se basa en el diseño general de los sistemas operativos Unix-like, en los que se busca una separación clara entre los componentes del sistema operativo y las aplicaciones de usuario, para mejorar la modularidad, la portabilidad y la mantenibilidad del sistema operativo.

En cuanto a la seguridad, la separación entre el kernel y el espacio de usuario se logra mediante el uso de mecanismos de protección de memoria y permisos de acceso que controlan el acceso de los programas de usuario a los recursos del sistema, incluyendo los recursos del kernel. De esta manera, se busca prevenir que un programa malicioso pueda acceder a recursos protegidos del sistema y causar daño o comprometer la seguridad.

(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 un intérprete de comandos distinto para cada usuario en GNU/Linux. Cada usuario tiene un archivo llamado ".bashrc" en su directorio de inicio ("home"), que es un archivo de configuración de Bash, el shell predeterminado en la mayoría de las distribuciones de GNU/Linux.

Este archivo permite personalizar el comportamiento del shell para cada usuario, incluyendo la definición de variables de entorno, alias, y funciones, así como la configuración de opciones y comportamientos específicos del shell.

Cualquier usuario con permisos de escritura en el archivo ".bashrc" de otro usuario puede editar este archivo y definir un intérprete de comandos diferente para ese usuario.

Sí, es posible, cada usuario puede tener una interfaz o shell. Esto se define al crear el usuario, o en archivo /etc/passwd, el archivo /etc/passwd solo puede ser modificado por un usuario root.

6. Sistema de Archivos (*File System*):

(a) ¿Qué es?

En un sistema operativo como GNU/Linux, el sistema de archivos es la estructura que organiza y gestiona los datos almacenados en el disco duro u otros dispositivos de almacenamiento.

En GNU/Linux, el sistema de archivos se organiza en una jerarquía de directorios, que comienza en el directorio raíz "/" y se extiende a través de varios subdirectorios y archivos en diferentes niveles. Cada directorio y archivo tiene su propia ruta única dentro del sistema de archivos.

Además, el sistema de archivos en GNU/Linux permite que los usuarios y aplicaciones accedan y gestionen archivos y directorios mediante comandos de línea de comandos o interfaces gráficas de usuario (GUI). El sistema de archivos también proporciona permisos y privilegios de acceso a archivos y directorios para garantizar la seguridad y la privacidad de los datos.

Existen varios sistemas de archivos en GNU/Linux, incluyendo ext2, ext3, ext4, XFS, Btrfs, entre otros, cada uno con características y capacidades diferentes. La elección del sistema de archivos a utilizar dependerá de las necesidades específicas de cada usuario o aplicación.

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

Existen varios sistemas de archivos soportados por GNU/Linux, algunos de los más utilizados son:

* Ext2: es el sistema de archivos por defecto en algunas distribuciones de Linux. Es un sistema de archivos antiguo, pero aún se utiliza en algunas distribuciones de Linux.
* Ext3: es una extensión del sistema de archivos Ext2 y fue diseñado para proporcionar journaling (registro de eventos en el sistema de archivos) para mejorar la fiabilidad del sistema de archivos.
* Ext4: es una extensión del sistema de archivos Ext3 y es el sistema de archivos predeterminado en la mayoría de las distribuciones de Linux modernas. Proporciona mejoras en la velocidad y la escalabilidad en comparación con Ext3.
* XFS: es un sistema de archivos de alto rendimiento que fue desarrollado originalmente por SGI (Silicon Graphics). Es especialmente adecuado para el almacenamiento en red y para grandes volúmenes de datos.
* Btrfs: es un sistema de archivos moderno y potente que proporciona funciones avanzadas como la compresión de datos, la instantánea de archivos y la duplicación de datos.

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

Sí, es posible visualizar particiones del tipo FAT y NTFS en GNU/Linux. De hecho, muchos sistemas Linux tienen soporte integrado para leer y escribir en particiones formateadas en sistemas de archivos FAT y NTFS.

Es importante tener en cuenta que, aunque se puede leer y escribir en particiones formateadas en sistemas de archivos FAT y NTFS, algunas funcionalidades avanzadas pueden no estar disponibles en estos sistemas de archivos en Linux.

(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**?

La estructura básica de los sistemas de archivos en GNU/Linux sigue la especificación del Sistema de Jerarquía de Archivos (FHS, por sus siglas en inglés), que define la organización de los directorios y archivos en el sistema operativo.

Los directorios más importantes y sus propósitos son los siguientes:

* / : es el directorio raíz del sistema de archivos y contiene todos los demás directorios y archivos en el sistema.
* /bin : contiene los programas esenciales del sistema que se requieren durante el arranque del sistema y para operaciones básicas del sistema.
* /boot : contiene los archivos del kernel y otros archivos necesarios para el arranque del sistema.
* /dev : contiene los archivos de dispositivo que representan los dispositivos de hardware del sistema.
* /etc : contiene los archivos de configuración del sistema, como archivos de configuración de red, de usuario y de servicios del sistema.
* /home : contiene los directorios de los usuarios del sistema.
* /lib : contiene las bibliotecas compartidas que utilizan los programas del sistema y algunos controladores de dispositivos.
* /mnt : es el directorio de montaje temporal para dispositivos de almacenamiento externos.
* /opt : es el directorio donde se instalan las aplicaciones de software de terceros.
* /proc : es un sistema de archivos virtual que proporciona información del sistema y del proceso en tiempo real.
* /root : es el directorio personal del usuario root.
* /sbin : contiene los programas esenciales del sistema para administradores de sistema.
* /tmp : es el directorio de almacenamiento temporal para archivos que se eliminan automáticamente al reiniciar el sistema.
* /usr : contiene los programas, bibliotecas, archivos de cabecera y documentación adicionales del sistema.
* /var : contiene los archivos de registro del sistema, archivos temporales y otros archivos que cambian frecuentemente durante la operación del sistema.

La especificación FHS (Filesystem Hierarchy Standard) es una guía para la organización y la estructura de los sistemas de archivos en sistemas operativos tipo Unix, incluyendo GNU/Linux. Proporciona una forma estandarizada de organizar el sistema de archivos para facilitar la portabilidad de software entre diferentes distribuciones de Linux.

7. Particiones:

(a) Definición. Tipos de particiones. Ventajas y Desventajas.

Una partición es una sección lógica del disco duro o de cualquier otro dispositivo de almacenamiento que se utiliza para separar y organizar los datos en unidades más pequeñas y manejables. Las particiones permiten tener diferentes sistemas de archivos en un mismo disco duro y también pueden mejorar el rendimiento y la seguridad del sistema.

Existen diferentes tipos de particiones, entre los que se incluyen:

1. Particiones primarias: son las particiones básicas en un disco duro y se utilizan para almacenar los datos y el sistema operativo. Un disco duro puede tener hasta cuatro particiones primarias.
2. Particiones extendidas: son una forma de crear particiones adicionales en un disco duro y permiten tener más de cuatro particiones en un disco duro. Dentro de una partición extendida se pueden crear particiones lógicas.
3. Particiones lógicas: son particiones creadas dentro de una partición extendida y se utilizan para separar y organizar los datos en unidades más pequeñas y manejables.
4. Particiones de arranque: son particiones especiales que contienen el cargador de arranque del sistema operativo y se utilizan para iniciar el sistema operativo.

Las ventajas de las particiones incluyen:

* Organización de los datos: las particiones permiten organizar los datos de forma más eficiente y separar los datos del sistema operativo.
* Mayor seguridad: las particiones pueden ayudar a proteger los datos de errores o daños en el sistema operativo o en otros datos.
* Mejora del rendimiento: las particiones pueden ayudar a mejorar el rendimiento del sistema al separar los datos del sistema operativo y de las aplicaciones.

Las desventajas de las particiones incluyen:

* Pérdida de espacio: cada partición utiliza una cierta cantidad de espacio en el disco duro, por lo que muchas particiones pueden llevar a una pérdida de espacio de almacenamiento.
* Dificultad para cambiar el tamaño: si se desea cambiar el tamaño de una partición, se puede requerir de herramientas adicionales y puede ser un proceso complicado.
* Riesgo de pérdida de datos: las particiones pueden aumentar el riesgo de pérdida de datos si no se realizan copias de seguridad de forma regular.

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

En GNU/Linux, las particiones se identifican mediante un esquema de nomenclatura basado en el dispositivo físico y la partición lógica. El sistema operativo asigna un nombre único a cada partición para que pueda ser identificada y montada correctamente.

Para los discos IDE, SCSI y SATA, los nombres de los dispositivos se basan en la ubicación física del dispositivo. Por ejemplo, para el primer disco duro IDE (hda) en el sistema, las particiones se nombran como hda1, hda2, hda3, etc. Para el segundo disco duro IDE (hdb), las particiones se nombran como hdb1, hdb2, hdb3, etc.

En GNU/Linux, un disco IDE puede ser nombrado utilizando la nomenclatura "/dev/hd[a-z]" , donde "a" y "b" representan el primer y segundo controlador IDE, respectivamente, y "z" representa el número de partición.

En el caso de los discos SCSI y SATA, el sistema utiliza el esquema de nomenclatura sda, sdb, sdc, etc., en lugar de hda, hdb, hdc, etc. Las particiones se nombran como sda1, sda2, sda3, etc., para la primera partición en el primer disco SCSI o SATA.

También es importante tener en cuenta que en los sistemas modernos de GNU/Linux, los discos pueden estar etiquetados por UUID (identificador único universal) en lugar de por su ubicación física. En este caso, la nomenclatura de las particiones puede incluir el UUID en lugar de la ubicación física. Por ejemplo, una partición en un disco con UUID "6b8b6f2b-2a80-4217-b6a8-6c585f6d4014" podría nombrarse como "/dev/disk/by-uuid/6b8b6f2b-2a80-4217-b6a8-6c585f6d4014".

(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 GNU/Linux, se requiere al menos una partición raíz ("root partition") y una partición de intercambio ("swap partition"). La partición raíz contendrá el sistema de archivos raíz y los directorios necesarios para el sistema operativo, mientras que la partición de intercambio se utiliza como espacio de almacenamiento temporal cuando la memoria RAM se agota.

A continuación, se muestra una configuración básica de particiones para instalar GNU/Linux:

* Partición raíz: Esta es la partición principal del sistema de archivos y contendrá todos los archivos del sistema operativo. Esta partición se puede identificar como "/dev/sda1" (o "/dev/hda1" en sistemas más antiguos) y se puede formatear con el sistema de archivos ext4. El punto de montaje de esta partición es "/".
* Partición de intercambio: Esta partición se utiliza como espacio de almacenamiento temporal cuando la memoria RAM se agota. Esta partición se puede identificar como "/dev/sda2" (o "/dev/hda2" en sistemas más antiguos) y se puede formatear con el sistema de archivos "swap". No se monta en ningún punto del sistema de archivos.

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

A continuación, se presentan algunos ejemplos de particionamiento según el tipo de virtualización a utilizar:

* Emulación: Si se está utilizando emulación, es decir, si se está emulando completamente una computadora dentro de otra, se puede utilizar un particionamiento similar al de una computadora física. Se puede configurar una partición raíz para el sistema operativo y programas, una partición de intercambio, y una partición para los datos de usuario. Además, se puede crear una partición separada para la imagen del disco virtual que se está utilizando para la emulación.

Permiten emular Hardware y tienen que implementar todas las instrucciones de la CPU.

* + Ventajas: Puede ejecutar sistemas operativos que no son compatibles con el hardware del anfitrión, lo que permite la compatibilidad con una amplia variedad de sistemas operativos. Además, puede emular hardware adicional, como tarjetas de red y dispositivos de almacenamiento, lo que permite una mayor flexibilidad en la configuración de la máquina virtual.
  + Desventajas: La emulación consume más recursos de la computadora anfitriona que otros métodos de virtualización, lo que puede resultar en una menor eficiencia y rendimiento. Además, la emulación es más lenta que otros métodos de virtualización, lo que puede afectar la velocidad de la máquina virtual.
* Virtualización completa: Si se está utilizando virtualización completa, donde se ejecuta un sistema operativo completo en una máquina virtual, se puede utilizar un particionamiento similar al de una computadora física. Se puede configurar una partición raíz para el sistema operativo y programas, una partición de intercambio, y una partición para los datos de usuario. Además, se puede crear una partición separada para la imagen de la máquina virtual y otra partición para los archivos de configuración y las instantáneas de la máquina virtual.
  + Ventajas: La virtualización completa es más eficiente que la emulación, ya que se ejecuta en el mismo hardware que el anfitrión, lo que permite un mejor rendimiento. Además, la virtualización completa es más rápida que la emulación, lo que puede mejorar la velocidad de la máquina virtual.
  + Desventajas: La virtualización completa no puede ejecutar sistemas operativos que no son compatibles con el hardware del anfitrión, lo que limita la flexibilidad en la selección de sistemas operativos. Además, la virtualización completa consume más recursos de la computadora anfitriona que otros métodos de virtualización, lo que puede resultar en una menor eficiencia y rendimiento.
* Paravirtualización: Si se está utilizando paravirtualización, donde se comparte el mismo kernel entre la máquina anfitriona y las máquinas virtuales, no es necesario un particionamiento complejo. En este caso, se puede utilizar una sola partición para el sistema operativo y los programas, y otra partición separada para los datos de usuario.
  + Ventajas: La paravirtualización es más eficiente que la emulación y la virtualización completa, ya que utiliza el mismo kernel que el anfitrión, lo que permite un mejor rendimiento. Además, la paravirtualización consume menos recursos de la computadora anfitriona que otros métodos de virtualización, lo que puede resultar en una mayor eficiencia y rendimiento.
  + Desventajas: La paravirtualización requiere que el sistema operativo de la máquina virtual sea compatible con la paravirtualización, lo que puede limitar la selección de sistemas operativos. Además, la paravirtualización no puede emular hardware adicional, como tarjetas de red y dispositivos de almacenamiento, lo que limita la flexibilidad en la configuración de la máquina virtual.

(e) ¿Qué tipo de software para particionar existe? Menciónelos y compare

Existen varios programas para particionar discos en GNU/Linux:

* GParted: es una herramienta gráfica para particionar discos en GNU/Linux. Es fácil de usar y cuenta con una interfaz de usuario intuitiva. Soporta una gran cantidad de sistemas de archivos, incluyendo ext2, ext3, ext4, NTFS, FAT32, entre otros.
* fdisk: es una herramienta de línea de comandos para particionar discos en GNU/Linux. Es una de las herramientas más antiguas y más utilizadas en GNU/Linux. Es muy poderosa, pero requiere un conocimiento técnico avanzado para utilizarla correctamente.
* parted: es una herramienta de línea de comandos para particionar discos en GNU/Linux. Es similar a fdisk, pero cuenta con algunas características adicionales, como la capacidad de redimensionar particiones sin perder datos.
* cfdisk: es una herramienta de línea de comandos para particionar discos en GNU/Linux. Es una alternativa más fácil de usar que fdisk y ofrece una interfaz de usuario más intuitiva.

Existen dos tipos de particionadores: los particionadores destructivos y los no destructivos

1. Particionadores destructivos: Los particionadores destructivos eliminan completamente las particiones existentes y crean nuevas. Esto significa que cualquier dato almacenado en las particiones existentes se perderá. Algunos ejemplos de particionadores destructivos son fdisk, cfdisk y parted.

Ventajas:

* 1. Permite una partición completamente nueva del disco.
  2. Los particionadores destructivos son más rápidos y más eficientes que los no destructivos.
  3. Son adecuados para usuarios que no necesitan guardar los datos de las particiones existentes.

Desventajas:

* 1. Se pierden todos los datos de las particiones existentes.
  2. Las particiones existentes no pueden ser recuperadas después de usar un particionador destructivo.

1. Particionadores no destructivos: Los particionadores no destructivos crean nuevas particiones sin borrar los datos existentes. Algunos ejemplos de particionadores no destructivos son GParted y KDE Partition Manager.

*Ventajas*:

* 1. Permite crear nuevas particiones sin perder los datos de las particiones existentes.
  2. Son adecuados para usuarios que necesitan conservar los datos existentes en las particiones.
  3. Las particiones existentes pueden ser recuperadas en caso de que se produzca un error en el proceso de particionamiento.

*Desventajas*:

* 1. Los particionadores no destructivos son más lentos que los destructivos.
  2. Pueden ser menos eficientes y menos precisos que los particionadores destructivos.

8. Arranque (*bootstrap*) de un Sistema Operativo:

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

El BIOS (Basic Input/Output System) es un firmware (un software almacenado en un chip de memoria no volátil) que se encuentra en la placa base de una computadora (Motherboard). Es el primer programa que se ejecuta cuando se enciende la computadora y es responsable de realizar varias tareas críticas en el proceso de arranque del sistema operativo.

Las principales tareas que realiza el BIOS son:

1. Comprobar el hardware: El BIOS realiza una serie de pruebas de diagnóstico para comprobar que el hardware del sistema está funcionando correctamente. Esta es la razón por la que a veces se muestra una pantalla con información sobre el hardware cuando se enciende una computadora.
2. Configurar los dispositivos de hardware: El BIOS configura los dispositivos de hardware de la computadora, como el disco duro, el teclado, el mouse y los puertos USB.
3. Cargar el sistema operativo: El BIOS busca en los dispositivos de almacenamiento conectados (por ejemplo, el disco duro, el CD-ROM, el USB) para encontrar el sistema operativo y cargarlo en la memoria RAM.
4. Transferir el control al sistema operativo: Una vez que se ha cargado el sistema operativo en la memoria RAM, el BIOS transfiere el control al sistema operativo para que pueda continuar con el proceso de arranque.

El **POST** (Power-On Self-Test) es una secuencia de diagnóstico que realiza el BIOS cuando se enciende una computadora. Su objetivo es verificar que el hardware del sistema esté funcionando correctamente antes de cargar el sistema operativo.

Durante el POST, el BIOS realiza varias pruebas en los componentes de hardware de la computadora, incluyendo la memoria RAM, el procesador, el disco duro, los dispositivos de almacenamiento y los dispositivos de entrada/salida como el teclado, el mouse y los puertos USB. El BIOS envía señales eléctricas a los componentes de hardware para verificar su respuesta y detectar posibles errores.

Si el POST detecta algún problema en el hardware, puede emitir pitidos o mostrar mensajes de error en la pantalla. Estos mensajes pueden indicar qué componente de hardware ha fallado y pueden ayudar al usuario a diagnosticar el problema.

Una vez completado el POST con éxito, el BIOS pasa al siguiente paso en el proceso de arranque: la búsqueda del sistema operativo.

(b) ¿Qué es **UEFI**? ¿Cuál es su función?

UEFI (Unified Extensible Firmware Interface) es un firmware que reemplaza al BIOS tradicional en las computadoras modernas. Es un software de bajo nivel que se ejecuta en el inicio del sistema y es responsable de inicializar el hardware del sistema y preparar el entorno para el sistema operativo.

La principal función de UEFI es proporcionar una interfaz entre el hardware de la computadora y el sistema operativo, y también proporcionar una serie de características avanzadas que no están disponibles en el BIOS. Entre estas características se incluyen:

* Soporte para discos duros de más de 2 TB.
* Arranque más rápido del sistema operativo.
* Mejora en la seguridad del arranque mediante la firma digital de los controladores y el firmware.
* Interfaz gráfica de usuario (GUI) con ratón y soporte táctil.
* Soporte para dispositivos de almacenamiento externos, como dispositivos USB y tarjetas SD.

UEFI también es más flexible que el BIOS en cuanto a la configuración del hardware del sistema. Proporciona una interfaz de configuración más avanzada que permite a los usuarios cambiar la configuración del hardware directamente desde una interfaz gráfica de usuario, sin necesidad de acceder a la BIOS mediante una tecla específica en el arranque.

(c) ¿Qué es el **MBR**? ¿Qué es el **MBC**?

El MBR (Master Boot Record) es una pequeña área del disco duro que se encuentra en el primer sector del disco y se utiliza para iniciar el sistema operativo en los sistemas BIOS. Ocupa exactamente 512 bytes (el tamaño de un sector en el disco duro). Esto se debe a que el MBR se encuentra en el primer sector del disco duro, el cual es el sector de arranque del disco.

El MBR contiene información importante sobre la estructura de partición del disco duro, así como un pequeño programa de arranque llamado Master Boot Code (MBC). El MBR también contiene una tabla de particiones que describe la ubicación y el tamaño de las particiones del disco duro.

Cuando se inicia el sistema, el firmware del BIOS busca el MBR en el primer sector del disco duro y carga el MBC en la memoria del sistema. A continuación, el MBC busca la partición activa y carga el sector de arranque del sistema operativo en la memoria del sistema para que se inicie el sistema.

El MBC (Master Boot Code) es una parte integral del MBR (Master Boot Record) y se encuentra en la sección de código de arranque del MBR. Es un pequeño programa de código máquina que se carga en la memoria del sistema cuando se inicia un sistema BIOS. Se encuentra en los últimos 446 bytes del MBR, es decir, desde el byte 446 al byte 511. Por lo tanto, el MBC ocupa un espacio de 66 bytes dentro del MBR.

El MBC es responsable de cargar el sector de arranque del sistema operativo desde la partición activa del disco duro. Cuando se enciende el sistema, el firmware del BIOS busca el MBR en el primer sector del disco duro y carga el MBC en la memoria del sistema. A continuación, el MBC busca la partición activa y carga el sector de arranque del sistema operativo en la memoria del sistema para que se inicie el sistema.

El MBC se encuentra en el sector de arranque del disco duro y es la primera cosa que se carga en la memoria del sistema cuando se inicia el sistema.

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

Las siglas GPT significan "GUID Partition Table" (Tabla de particiones con identificador único global). GPT es un tipo de particionamiento de discos duros y otros dispositivos de almacenamiento de datos que reemplaza al antiguo esquema de particionamiento de disco MBR (Master Boot Record).

GPT se utiliza en sistemas que usan la interfaz UEFI (Unified Extensible Firmware Interface) en lugar del BIOS tradicional. A diferencia de MBR, que solo admite hasta 4 particiones primarias, GPT permite la creación de hasta 128 particiones en un solo disco.

El formato de GPT es una tabla de particiones que se encuentra en el sector de arranque protegido (PMBR) de un disco. La tabla de particiones contiene información sobre la estructura de particiones en el disco, incluyendo el tamaño, la ubicación y los tipos de particiones. GPT también incluye una copia de seguridad de la tabla de particiones en la parte final del disco para proteger contra fallas en la tabla de particiones principal.

En GPT (GUID Partition Table), el direccionamiento lógico se refiere a la forma en que se identifican y direccionan las particiones dentro del disco. En lugar de utilizar la numeración de particiones tradicional utilizada por MBR (Master Boot Record), que asigna números de partición como 1, 2, 3, etc., GPT utiliza identificadores únicos globales (GUID) para cada partición. Esto significa que cada partición en el disco tiene un identificador único y no se asignan números fijos a las particiones.

En cuanto al MBR, utiliza un tipo de direccionamiento conocido como direccionamiento basado en cilindros, cabezas y sectores (CHS). Este método de direccionamiento utiliza números enteros para indicar la posición de una partición en el disco en relación con la cabeza del disco, el número del cilindro y el sector en el que se encuentra. Sin embargo, este método de direccionamiento está limitado por la capacidad del disco y la cantidad de cabezas y cilindros que pueden ser direccionados.

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

Un gestor de arranque, también conocido como bootloader, es un programa que se encarga de cargar el sistema operativo en la memoria del ordenador y transferir el control al kernel del sistema operativo para que se inicie el proceso de arranque.

Existen varios tipos de gestores de arranque, pero los dos más comunes son GRUB (GNU GRand Unified Bootloader) y el gestor de arranque de Windows. GRUB es utilizado por la mayoría de las distribuciones de Linux, mientras que el gestor de arranque de Windows se utiliza exclusivamente en sistemas operativos Windows.

Los gestores de arranque se instalan en el sector de arranque maestro (MBR) del disco duro o en la partición de arranque (Volume Boot Record). Cuando el equipo arranca, el BIOS o UEFI carga el gestor de arranque desde el sector de arranque maestro y luego el gestor de arranque carga el sistema operativo.

Algunos gestores de arranque conocidos son:

* GRUB (GNU GRand Unified Bootloader): Es el gestor de arranque predeterminado en la mayoría de las distribuciones de Linux. Es altamente configurable y permite arrancar varios sistemas operativos y kernels.
* LILO (Linux Loader): Es un gestor de arranque para Linux que ha sido superado por GRUB, pero aún es utilizado en algunas distribuciones antiguas.
* Windows Boot Manager: Es el gestor de arranque utilizado en los sistemas operativos Windows. Se instala en la partición de arranque y permite arrancar varios sistemas operativos de Windows.
* Syslinux: Es un gestor de arranque para sistemas basados en Linux que se utiliza en sistemas embebidos y dispositivos de almacenamiento USB.
* rEFInd: Es un gestor de arranque para sistemas basados en UEFI que se utiliza en sistemas MacOS y Linux. Es altamente personalizable y permite arrancar varios sistemas operativos y kernels.

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

Estos pasos pueden variar dependiendo del tipo de firmware y del sistema operativo, pero en general los pasos son los siguientes:

1. Encendido de la computadora: al presionar el botón de encendido, se suministra energía eléctrica al sistema y se inicia el proceso de arranque.
2. POST (Power-On Self-Test): el firmware (BIOS o UEFI) realiza un autodiagnóstico de hardware para verificar que todos los componentes estén funcionando correctamente. Si el POST detecta algún problema, se emiten pitidos de alerta y se muestra un mensaje en la pantalla.
3. Inicio del gestor de arranque: el firmware carga el gestor de arranque (GRUB, LILO, Windows Boot Manager, etc.) desde el sector de arranque del disco duro o desde una partición especial de arranque.
4. Selección del sistema operativo: el gestor de arranque muestra una lista de los sistemas operativos instalados y permite al usuario seleccionar cuál desea arrancar.
5. Carga del sistema operativo: el gestor de arranque carga el kernel del sistema operativo seleccionado en la memoria RAM y lo ejecuta. A partir de este punto, el sistema operativo se encarga de realizar la inicialización del hardware y de cargar los servicios y aplicaciones necesarias.
6. Inicio de sesión: cuando el sistema operativo ha terminado de cargar, se muestra la pantalla de inicio de sesión, donde el usuario debe ingresar sus credenciales para acceder al sistema.

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

El proceso de arranque en GNU/Linux consta de varios pasos que se llevan a cabo desde que se enciende el equipo hasta que el sistema operativo está completamente cargado y listo para su uso. A continuación, se describen los principales pasos del proceso de arranque en GNU/Linux:

1. Encendido del equipo: al encender el equipo, la BIOS o UEFI realiza las pruebas de autoevaluación (POST) para verificar el estado de los componentes del hardware.
2. Carga del cargador de arranque: después de la autoevaluación, se carga el cargador de arranque, que es el encargado de iniciar el sistema operativo. El cargador de arranque más utilizado en GNU/Linux es GRUB (GNU GRand Unified Bootloader).
3. Carga del kernel de Linux: una vez que se ha cargado el cargador de arranque, este carga el kernel de Linux en la memoria RAM. El kernel es el núcleo del sistema operativo y se encarga de gestionar los recursos del hardware, así como de proporcionar las funciones básicas del sistema.
4. Inicio del proceso "init": después de cargar el kernel, se inicia el proceso "init", que es el primer proceso que se ejecuta en el sistema y es responsable de iniciar todos los demás procesos del sistema operativo. El proceso "init" también es el encargado de gestionar el nivel de ejecución del sistema, que determina qué servicios se deben iniciar.
5. Arranque del entorno gráfico: en sistemas que cuentan con entorno gráfico, el proceso "init" inicia el gestor de ventanas (como GDM o LightDM), que es el encargado de proporcionar la interfaz gráfica de usuario. Después de que se inicia el gestor de ventanas, se carga el escritorio del usuario (como GNOME o KDE).

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

El proceso de parada (shutdown) en GNU/Linux sigue los siguientes pasos:

1. El usuario inicia el comando de parada (shutdown) desde la terminal o desde una interfaz gráfica.
2. El comando de parada (shutdown) avisa a todos los usuarios conectados que el sistema se va a apagar y que deben guardar su trabajo y cerrar sesión.
3. Se detienen los servicios del sistema, incluyendo los que están en segundo plano y los que se están ejecutando en ese momento.
4. Se desmontan los sistemas de archivos para garantizar que todos los datos se han escrito en el disco y que no se perderán.
5. Se envía una señal a todos los procesos para que finalicen su ejecución y guarden su estado actual.
6. Se sincronizan los sistemas de archivos y se escriben en el disco todos los datos pendientes.
7. Se apagan los dispositivos de hardware y se detiene el procesador.

El proceso de parada (shutdown) puede ser de varios tipos, dependiendo de la opción que se seleccione al ejecutar el comando. Las opciones más comunes son:

* Shutdown inmediato (shutdown now): apaga el sistema de inmediato.
* Shutdown programado (shutdown + tiempo): apaga el sistema después de un tiempo determinado.
* Reinicio del sistema (reboot): reinicia el sistema en lugar de apagarlo.

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

Sí, es posible tener en una PC instalado GNU/Linux y otro Sistema Operativo (como Windows, macOS, FreeBSD, entre otros) de manera dual, es decir, ambos sistemas operativos pueden coexistir en el mismo equipo y ser elegidos por el usuario al momento de arrancar la computadora. Esto se logra mediante la creación de particiones separadas para cada sistema operativo y la instalación de un gestor de arranque (como GRUB o LILO) que permita seleccionar el sistema operativo a iniciar.

Esta configuración puede resultar útil para aquellos usuarios que necesiten utilizar diferentes sistemas operativos para distintas tareas o para tener acceso a programas o aplicaciones específicas que solo estén disponibles en uno de los sistemas operativos. Además, la posibilidad de tener ambos sistemas operativos instalados en el mismo equipo permite una mayor flexibilidad y versatilidad en el uso de la computadora.

9. Archivos:

(a) ¿Cómo se identifican los archivos en *GNU/Linux*?

En GNU/Linux, los archivos se identifican mediante su nombre y su ruta en el sistema de archivos. El nombre de un archivo puede contener letras, números, caracteres especiales y espacios en blanco. Los nombres de archivo distinguen entre mayúsculas y minúsculas, lo que significa que "Archivo.txt" y "archivo.txt" son dos archivos diferentes. No pueden utilizar la barra “/”.

La ruta de un archivo en el sistema de archivos es una secuencia de nombres de directorios separados por barras (/), que indica la ubicación del archivo en el árbol de directorios. La ruta puede ser absoluta, comenzando desde la raíz del sistema de archivos (/), o relativa, comenzando desde el directorio actual.

Además del nombre y la ruta, los archivos en GNU/Linux se identifican mediante permisos, propietarios y grupos. Estos atributos determinan quién puede acceder y modificar los archivos. Los permisos se establecen mediante una serie de bits que indican si el archivo puede ser leído, escrito o ejecutado por el propietario, el grupo y otros usuarios. El propietario y el grupo de un archivo se especifican mediante sus identificadores numéricos (UID y GID) o sus nombres de usuario y grupo correspondientes.

(b) Investigue el funcionamiento de los editores **vi** y **mcedit**, y los comandos **cat** y **more**.

Los editores de texto vi y mcedit son dos de los editores de texto más populares en sistemas Unix y Unix-like, incluyendo GNU/Linux. El editor vi (visual) es un editor de texto basado en línea de comandos que se ejecuta en la terminal. Por otro lado, mcedit es un editor de texto basado en menús y ventanas, que también se ejecuta en la terminal.

El editor vi es un editor de texto que maneja en memoria el texto entero de un archivo. Es el editor clásico de UNIX (se encuentra en todas las versiones).

El editor vi se compone de dos modos principales: el modo de comando y el modo de inserción. En el modo de comando, las teclas presionadas tienen significado como comandos, y permiten mover el cursor, copiar y pegar texto, eliminar líneas, guardar el archivo, entre otras acciones. En el modo de inserción, las teclas presionadas insertan texto en el archivo.

El editor mcedit también se compone de un modo de edición y un modo de comando. En el modo de edición, se pueden escribir y editar texto usando atajos de teclado y menús. En el modo de comando, se pueden ejecutar comandos para buscar y reemplazar texto, guardar el archivo, entre otras acciones.

En muchos sistemas operativos tipo Unix, el comando mcedit es un enlace simbólico al editor interno de archivos del Midnight Commander (mc), que es un administrador de archivos de texto basado en consola. Por lo tanto, mcedit es una especie de extensión del Midnight Commander que permite editar archivos de texto directamente en el mismo entorno de consola donde se ejecuta el administrador de archivos.

El comando cat se utiliza para concatenar y mostrar el contenido de uno o más archivos de texto en la terminal.

El comando more es similar a cat, pero muestra el contenido de un archivo de texto de forma paginada. Es decir, se muestra una pantalla de texto a la vez, y se puede avanzar o retroceder en el archivo utilizando las teclas de flecha y la barra espaciadora. También admite la búsqueda de cadenas o expresiones regulares y la apertura del archivo en el punto actual en un editor de texto.

(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.

Para crear un archivo llamado "prueba.exe" con tu nombre y legajo utilizando el vi, hay que realizar los siguientes pasos:

1. Abrir la terminal y escribir el comando "vi prueba.exe" y presiona Enter.
2. Aparecerá una pantalla en blanco, presiona la tecla "i" para entrar en modo de inserción.
3. Escribe tu número de alumno y nombre en la pantalla.
4. Presiona la tecla "Esc" para salir del modo de inserción.
5. Escribe ":wq" y presiona Enter para guardar el archivo y salir del vi.

Ahora tendrás un archivo llamado "prueba.exe" con tu nombre "Nombre" creado con el vi. Cabe destacar que el archivo no tendrá la extensión ".exe" como en sistemas operativos Windows, ya que esta extensión se utiliza principalmente en ese sistema operativo para identificar archivos ejecutables. En GNU/Linux no es necesario utilizar extensiones de archivo para identificarlos.

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

El comando file es una herramienta en línea de comando utilizada en sistemas operativos tipo Unix para identificar el tipo de archivo de un archivo determinado. El comando file utiliza una serie de heurísticas para examinar los contenidos de un archivo y determinar el tipo de archivo. Esto se hace comparando la estructura y el contenido del archivo con una base de datos de firmas de archivo predefinidas.

Probando con diferentes archivos, se puede notar que file es capaz de identificar el tipo de archivo, incluyendo información como el formato, la extensión, la codificación y otros detalles relevantes para ese tipo de archivo. También puede ser utilizado para verificar si un archivo es binario o de texto plano, lo que puede ser útil en situaciones donde se necesita analizar un archivo para determinar su tipo de contenido.

La invocación del comando file tiene el siguiente formato:

* **file [PARÁMETROS] *ARCHIVO*...**

Los parámetros posibles, según la especificación [POSIX](https://es.wikipedia.org/wiki/POSIX), son:

**-d →** Realiza las pruebas de sintaxis y de números mágicos del sistema. Esta es la opción default, sino se utiliza *-m* o *-M*

**-h →** Si el archivo a analizar es un enlace simbólico, lo identifica como tal.

**-i →** No brindar una clasificación adicional más allá de la del tipo básico.

**-m *archivo →*** Realiza una prueba adicional de números mágicos con el archivo indicado

**-M *archivo →*** Similar a *-m*, salvo que no realiza las pruebas de sintaxis y de números mágicos por defecto del sistema.

En los sistemas operativos [GNU/Linux](https://es.wikipedia.org/wiki/GNU/Linux), este comando fue enriquecido con muchos más parámetros, entre los cuales se encuentran:

**-b →** No imprime el nombre del archivo en cada resultado.

**-i →** Muestra el tipo mime junto con la codificación utilizada.

**-e *prueba →*** Excluye de realizar la prueba indicada. Entre las opciones que podemos indicarle, se encuentran:

**apptype**

Tipo de aplicación EMX (solo para EMX).

**ascii**

Varios tipos de archivos de texto. Esta prueba intenta determinar la codificación, más allá de la indicada dentro del propio archivo.

**encoding**

Varios tipos de codificaciones para la prueba suave de números mágicos.

**tokens**

Busca cadenas conocidas dentro de los archivos de texto.

**cdf**

Muestra detalles de los archivos CDF (Compound Document Files). POr ejemplo SVG, XHTML, etc.

**compress**

Analiza y busca dentro de los archivos comprimidos.

**elf**

Muestra detalles de archivos ELF.

**soft**

Consulta de archivos mágicos.

**tar**

Analiza archivos [Tar](https://es.wikipedia.org/wiki/Tar).

**--mime-type →** similar a *-i*, salvo porque solo muestra el tipo mime.

**-z →** Examina los archivos comprimidos.

10. 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

mkdir: crea un directorio en la ubicación especificada. Los parámetros más importantes son:

* -p: crea directorios padres si no existen
* -m: establece los permisos del directorio
* --verbose. Muestra un mensaje para cada directorio creado.

**mkdir** ISO2017

(b) Acceda a la carpeta (cd)

cd: cambia el directorio actual. Los parámetros más importantes son:

* ~: cambia al directorio personal del usuario
* ..: cambia al directorio padre del directorio actual

**cd** ISO2017

(c) Cree dos archivos con los nombres iso2017-1 e iso2017-2 (touch)

touch: crea un archivo vacío en la ubicación especificada. Los parámetros más importantes son:

* -a: actualiza la fecha de acceso del archivo
* -m: actualiza la fecha de modificación del archivo
* -t: establece una fecha y hora específicas para el archivo
* -c= No crea archivos que no existían antes.

**touch** iso2017-1 iso2017-2

(d) Liste el contenido del directorio actual (ls)

ls: lista el contenido del directorio actual. Los parámetros más importantes son:

* -l: lista los archivos y directorios en formato de lista
* -a: muestra todos los archivos, incluyendo los ocultos
* -h: muestra los tamaños de archivo en formato legible por humanos
* -t: ordena los archivos por fecha de modificación
* -X: ordena los archivos por extensión.
* -R: Muestra el contenido de todos los subdirectorios de forma recursiva
* -S: Ordena los resultados por tamaño de archivo

**ls**

(e) Visualizar la ruta donde estoy situado (pwd)

pwd: muestra la ruta del directorio actual.

**pwd**

(f) Busque todos los archivos en los que su nombre contiene la cadena “iso\*” (find)

find: busca archivos y directorios en la ubicación especificada. Los parámetros más importantes son:

* -name: busca archivos con un nombre específico
* -iname: busca archivos con un nombre específico sin tomar en cuenta mayúsculas y minúsculas
* -type: busca archivos de un tipo específico (por ejemplo, directorios, archivos regulares, enlaces simbólicos)
* -mtime: busca archivos modificados en un período de tiempo específico

**find** -name "iso\*"

Otras opciones:

Busca desde raiz archivos que tengan de nombre "reporte"

* find / -name reporte

El uso de expresiones regulares en lo que se busca es válido:

* find / -name "[0-9]\*" (todo lo que empieze con un dígito)
* find / -name "[Mm]\*" (todo lo que empieze con un la letra M o m)
* find / -name "[a-m]\*.txt" (todo lo que empieze entre a y m y termine en ".txt")

Busca bajo /home todos los archivos que pertenezcan al usuario mario:

* find /home -user mario

No estás limitado a un solo directorio, indica más de uno a buscar antes de las expresiones:

* find /etc /usr /var -group admin (busca en tres directorios todos los archivos o subdirectorios que pertenezcan al grupo 'admin')

(g) Informar la cantidad de espacio libre en disco (df)

df: muestra el espacio libre en disco. Los parámetros más importantes son:

* -h: muestra los tamaños de almacenamiento en formato legible por humanos
* -T: muestra el tipo de sistema de archivos

**df** -h

(h) Verifique los usuarios conectados al sistema (who)

who: muestra los usuarios conectados al sistema. Los parámetros más importantes son:

* -a: muestra información adicional sobre los usuarios, como la última vez que iniciaron sesión y la dirección IP

**who**

(i) Acceder a el archivo iso2017-1 e ingresar Nombre y Apellido

vi: es un editor de texto en línea de comandos. Los parámetros más importantes son:

* -n: muestra el número de línea en la barra lateral
* -i: activa la compatibilidad con el editor de texto "vi" original
* -R: abre el archivo en modo de solo lectura

vi iso2017-1

Luego, se presiona "i" para ingresar al modo de inserción, se escribe el Nombre y Apellido y se presiona "Esc" para salir del modo de inserción. Finalmente, se escribe ":wq" para guardar y salir del archivo.

(j) Mostrar en pantalla las últimas líneas de un archivo (tail).

tail: muestra las últimas líneas de un archivo. Los parámetros más importantes son:

* -n: especifica el número de líneas para mostrar
* -f: sigue el archivo en tiempo real y muestra nuevas líneas agregadas.

**tail** archivo.txt

11. Investigue su funcionamiento y parámetros más importantes:

(a) shutdown: Este comando se utiliza para apagar o reiniciar el sistema. Algunos de los parámetros más importantes son:

* -h o --halt: Apaga el sistema.
* -r o --reboot: Reinicia el sistema.
* -c o --cancel: Cancela el apagado o reinicio programado.

(b) reboot: Este comando reinicia el sistema. Algunos de los parámetros más importantes son:

* -f o --force: Reinicia el sistema sin realizar las comprobaciones normales.
* -n o --no-sync: No sincroniza el sistema de archivos antes de reiniciar.

(c) halt: Este comando apaga el sistema. Algunos de los parámetros más importantes son:

* -f o --force: Apaga el sistema sin realizar las comprobaciones normales.

(d) locate: Este comando busca archivos en el sistema. Algunos de los parámetros más importantes son:

* -i o --ignore-case: Realiza la búsqueda sin tener en cuenta las mayúsculas o minúsculas.
* -r o --regexp: Utiliza una expresión regular para realizar la búsqueda.
* -c hace un conteo de cuantos contienen la palabra pasada por argumento
* -e comprueba que el archivo aun exista en el sistema

(e) uname: Este comando muestra información sobre el sistema. Algunos de los parámetros más importantes son:

* -a o --all: Muestra toda la información disponible.
* -r o --kernel-release: Muestra la versión del kernel.
* -v: Muestra la fecha de cuando la versión del kernel fue liberada
* -p: Muestra información del procesador
* -o: Muestra información de la distribución de Linux que estamos utilizando

(f) gmesg: Este comando muestra los mensajes del kernel. Algunos de los parámetros más importantes son:

* -c o --clear: Borra el búfer de mensajes del kernel.
* -T o --ctime: Muestra la hora de creación de los mensajes.

(g) lspci: Este comando muestra información sobre los dispositivos PCI conectados al sistema. Algunos de los parámetros más importantes son:

* -v o --verbose: Muestra información detallada sobre los dispositivos.
* -n o --numeric: Muestra los identificadores numéricos de los dispositivos.

(h) at: Este comando programa la ejecución de un comando en una hora determinada. Algunos de los parámetros más importantes son:

* -f o --file: Ejecuta el comando desde un archivo.
* -l o --list: Muestra las tareas programadas.

(i) netstat: Este comando muestra información sobre la red. Algunos de los parámetros más importantes son:

* -a o --all: Muestra todas las conexiones.
* -n o --numeric: Muestra las direcciones IP y los números de puerto en formato numérico.

(j) mount: Este comando monta un sistema de archivos. Algunos de los parámetros más importantes son:

* -t o --type: Especifica el tipo de sistema de archivos.
* -o o --options: Especifica las opciones de montaje.

(k) umount: Este comando desmonta un sistema de archivos. Algunos de los parámetros más importantes son:

* -f o --force: Desmonta el sistema de archivos de forma forzada.
* -l o --lazy: Desmonta el sistema de archivos de forma perezosa.

(l) head: Este comando muestra las primeras líneas de un archivo. Algunos de los parámetros más importantes son:

* -n o --lines: Especifica el número de líneas a mostrar.
* -v o --verbose: Muestra información adicional.

(m) losetup: Este comando configura y muestra información sobre los dispositivos de bucle (loop devices). Algunos de los parámetros más importantes son:

* -f o --find: Asigna el primer dispositivo de bucle libre disponible.
* -d o --detach: Libera el dispositivo de bucle.

(n) write: Este comando permite enviar mensajes a otros usuarios conectados al sistema. Algunos de los parámetros más importantes son:

* <usuario>: Especifica el usuario al que se enviará el mensaje.
* <terminal>: Especifica el terminal al que se enviará el mensaje.

(ñ) mkfs: Este comando crea un sistema de archivos en un dispositivo de almacenamiento. Algunos de los parámetros más importantes son:

* -t o --type: Especifica el tipo de sistema de archivos a crear.
* -n o --label: Asigna una etiqueta al sistema de archivos.

(o) fdisk (con cuidado): Este comando permite crear, eliminar y modificar las particiones de un disco duro. Algunos de los parámetros más importantes son:

* -l o --list: Muestra la lista de discos y particiones.
* -t o --type: Cambia el tipo de partición.

12. Investigue su funcionamiento y parámetros más importantes:

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

En GNU/Linux, los comandos se almacenan en diferentes directorios del sistema:

* (a) **shutdown**, (b) **reboot**, (c) **halt**: Estos comandos generalmente se encuentran en el directorio /sbin/.
* (d) **locate**, (f) **gmesg**, (g) **lspci**, (h) **at**, (i) **netstat**, (j) **mount**, (k) **umount**, (l) **head**, (m) **losetup**, (n) **write**, (ñ) **mkfs**, (o) **fdisk**: Estos comandos generalmente se encuentran en el directorio /bin/.
* (e) **uname**: Este comando generalmente se encuentra en el directorio /usr/bin/.