1. Características de GNU/Linux:

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

GNU/Linux al ser un software libre posee las siguientes características:

- Puede ser usado, copiado, estudiado, modificado y redistribuido libremente.
- Generalmente es de costo nulo ← Es un gran error asociar el software libre con el software gratuito ← Pensar en software gratis que se distribuye con restricciones
- Es común que se distribuya junto con su código fuente
- Corrección más rápida ante fallas
- Características que se refieren a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software

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

Otros sistemas operativos como Windows o Mac Os al ser de software propietario:

- Generalmente tiene un costo asociado
- No se lo puede distribuir libremente
- Generalmente no permite su modificación
- Normalmente no se distribuye junto con su código fuente
- La corrección de fallas está a cargo del propietario (se deben esperar actualizaciones)
- Menos necesidad de técnicos especializados

(c) ¿Qué es GNU?

GNU = GNU No es Unix es el sistema operativo similar a Unix, constituido en su totalidad por software libre. El sistema GNU incluye todo el software GNU, además de muchos otros paquetes.

UNIX es un Sistema Operativo no libre muy popular, basado en una arquitectura estable. El sistema GNU fue diseñado para ser compatible con UNIX, haciendo que pueda ser compuesto de pequeñas piezas individuales de software que ya estaban disponibles (que pudieron ser adaptadas y reutilizadas).

Para asegurar que el software GNU permaneciera libre el proyecto debía ser liberado bajo una licencia diseñada para garantizar esos derechos al tiempo que evitase restricciones posteriores de los mismos. La idea se conoce como copyleft, y está contenida en la Licencia General Pública de GNU (GPL).

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

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 hace uso de la multitarea, multiusuario y multiprocesador. Que sea multitarea significa que le permite al usuario estar realizando varias tareas al mismo tiempo. Este puede estar editando el código fuente de un programa durante su depuración mientras compila otro programa, a la vez que está recibiendo correo electrónico en un proceso en background.

(f) ¿Qué es POSIX?

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 4 distribuciones de GNU/Linux y cite diferencias básicas entre ellas.

Son un conjunto de aplicaciones reunidas que permiten brindar mejoras para instalar fácilmente un sistema operativo basado en GNU/Linux. Son "sabores" de GNU/Linux que, en general, se diferencian entre sí por las herramientas para configuración y sistemas de administración de paquetes de software para instalar. La elección de una distribución depende de las necesidades del usuario y de gustos personales.

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, Madriva o la propia Ubuntu, y otras mantenidas por la comunidad Linux, como Debian, que no está relacionada con ninguna empresa y utiliza únicamente software.

	Logo	Productor	Existe Desde	Basado En	Ultima Version Estable	Escritorios que Soporta	Paquetes que Maneja
Debian		Debian Project	1993	----------	6.0 (Squeeze) Febrero 2011	Gnome,KDE, Xfce,Lxde.	deb
Fedora		Fedora Project	2003	Red Hat Linux	14 (Laughlin) Noviembre 2010	Gnome,KDE, Xfce.	rpm
Linux Mint		Comunidad	2006	Ubuntu	10 (Julia) Noviembre 2010	Gnome,KDE, Xfce.	deb
Mandriva	 7	Mandriva S.A	1998	Red Hat Linux	2010 (Farman) Marzo 2010	Gnome,KDE.	rpm
OpenSuse		Novell	1994	Suse Linux	11.4 Marzo 2011	Gnome,KDE, Xfce	rpm
Slackware		Comunidad	1993	----------	13.37 Abril 2011	KDE,Xfce.	tgz
Ubuntu		Canonical Ltda	2004	Debian	11.04 (Natty Narwall) Abril 2011	Gnome,KDE,Xfce,Lxde	deb

Las principales distribuciones GNU/Linux:

- Debian: Es una distribución de GNU/Linux realizada por una comunidad de desarrolladores y usuarios. Debian es la distribución de Linux más estable, que llega a un lema del sistema operativo libre y ungran conjunto de software gratuito para todos. Debian no se enfoca en los nuevos lanzamientos con frecuencia como Ubuntu y Linux Mint, pero se enfoca principalmente en un lanzamiento super estable. Por esa razón, Debian lanza una versión estable cada 2 años. Debian es básicamente una distribución que se usa para hacer muchas distribuciones populares y efectivas, como Ubuntu, Linux Mint, Deepin,Elementary OS, etc.
- Ubuntu: Está orientado al usuario promedio, con un fuerte enfoque en la facilidad de uso y en mejorar la experiencia del usuario. Ubuntu es uno de los más populares, estables y mejor equipados de las distribuciones Linux basadas en Debian. Tiene sus propios repositoriosde software que se sincronizan regularmente con el repositorio de Debian para que todas las aplicaciones se estabilicen y lancen la última versión. Ubuntu es desarrollado por Canonical
- Linux Mint: Linux Mint es la distribución basada en Ubuntu más popular y fácil de usar disponible en el mercado. Linux Mint es igualmente perfecto tanto para los recién llegados como para los usuarios avanzados. El lema principal de Linux Mint es “De la libertad vino la elegancia”, que proporciona una experiencia estable, poderosa,fácil de usar y completa. Como Linux Mint es una distribución de Linux basada en Ubuntu, será totalmente compatible con los repositorios de software de Ubuntu.
- Fedora: Distribución para propósitos generales, que se caracteriza por ser estable y seguro, la cual es desarrollada y mantenida por la empresa Red Hat y una comunidad internacional de ingenieros, diseñadores gráficos y usuarios que informan de fallos y prueban nuevas tecnologías. Sus usos se orientan más al desarrollo de software y servidores.

- Android: es un sistema operativo móvil desarrollado por Google, basado en el Kernel de Linux y otros software de código abierto. Fue diseñado para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tabletas, relojes inteligentes, automóviles y televisores.

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

Las distribuciones de Linux tienen en común el kernel, pero el resto de componentes (las herramientas, la shell, el Display Server, la GUI) varían entre sí, se personalizan o se crean desde cero, por eso las distribuciones son tan diferentes entre sí. Aunque en la mayoría de los casos la principal diferencia es la GUI, o los programas y herramientas que vienen incluidos.

Linux posee un kernel monolítico basado en Unix, multitarea apropiativa, memoria virtual, librerías compartidas.

Basado en que todo es un archivo (dispositivos, aplicaciones, etc). todo en unsistema Linux es un archivo, tanto el Software como el Hardware. Desde el ratón, pasando por la impresora, el reproductor de DVD, el monitor, un directorio, un subdirectorio y un fichero de texto.

Originariamente, en los inicios de Linux, este árbol de directorios no seguía un estándar cien por cien, es decir, podíamos encontrar diferencias en él de una distribución a otra.

Todo esto hizo pensar a cierta gente* que, posteriormente, desarrollarían el proyecto FHS (Filesystem Hierarchy Standard, o lo que es lo mismo: Estándar de Jerarquía de Sistema de Ficheros) en otoño de 1993. FHS se define como un estándar que detalla los nombres, ubicaciones, contenidos y permisos de los archivos y directorios, es decir, un conjunto de reglas que especifican una distribución común de los directorios y archivos en sistemas Linux.

Sistema monolítico (intercambia entre modo usuario y modo kernel cuando alguna tarea necesita alguna función de la que se tiene que hacer cargo el sistema operativo. El sistema operativo gestiona las tareas más importantes como control de acceso a memoria, entrada/salida dispositivos,etc)

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

El Proyecto Debian es una asociación de personas que han hecho causa común para crear un sistema operativo (SO) libre. Este sistema operativo que hemos creado se llama Debian.

Debian lo producen cerca de un millar de desarrolladores activos, dispersos por el mundo que ayudan voluntariamente en su tiempo libre. Son pocos los desarrolladores que realmente se han encontrado en persona. La comunicación se realiza principalmente a través de correo electrónico (listas de correo en lists.debian.org) y a través de IRC (canal #debian en irc.debian.org).

Debian 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

El nombre tiene su origen en los nombres del creador de Debian, Ian Murdock, y su esposa, Debra. Debian ha tenido varios líderes desde sus comienzos en el año 1993.

Debian ha tenido varios líderes desde sus comienzos en el año 1993.

Ian Murdock fundó Debian en agosto de 1993 y lo condujo hasta marzo de 1996.

Bruce Perens condujo Debian desde abril de 1996 hasta diciembre de 1997.

Ian Jackson condujo Debian desde enero de 1998 hasta diciembre de 1998.

Wichert Akkerman condujo Debian desde enero de 1999 hasta marzo de 2001.

Ben Collins condujo Debian desde abril de 2001 hasta abril de 2002.

Bdale Garbee condujo Debian desde abril de 2002 hasta abril de 2003.

Martin Michlmayr condujo Debian desde marzo de 2003 hasta marzo de 2005.

Branden Robinson condujo Debian desde abril de 2005 hasta abril de 2006.

Anthony Towns condujo Debian desde abril de 2006 hasta abril de 2007.

Sam Hocevar condujo Debian desde abril de 2007 hasta abril de 2008.

Steve McIntyre condujo Debian desde abril de 2008 hasta abril de 2010.

Stefano Zacchiroli condujo Debian desde abril de 2010 hasta abril de 2013.

Lucas Nussbaum condujo Debian desde abril de 2013 hasta abril de 2015.

Neil McGovern condujo Debian desde abril de 2015 hasta abril de 2016.

Mehdi Dogguy condujo Debian desde abril de 2016 hasta abril de 2017.

Chris Lamb condujo Debian desde abril 2017 hasta 2019.

Sam Hartman condujo Debian desde April 2019 desde April 2020.

Jonathan Carter fue elegido April 2020 and es el actual conductor.

ARBOL DE LA FAMILIA DEBIAN GNU/LINUX

3. Estructura de GNU/Linux:

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

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

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

La estructura básica del S.O es el Kernel, que ejecuta programas y gestiona dispositivos de hardware. Sus funciones más importantes son la administración de memoria, CPU y la E/S. El Kernel en un sentido estricto, es el sistema operativo.

4. Kernel:

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

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[1]. 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 otalmente compatible con 2.6
- La última versión estable es la 4.7.1 (agosto de 2016)

[1] : A.B.C[.D]

A: Denota versión. Cambia con menor frecuencia.

B: Denota mayor revisión.

C: Denota menor revisión. Solo cambia cuando hay nuevos drivers o características.

D: Cambia cuando se corrige un grave error sin agregar nueva funcionalidad ← Casi no se usa en las ramas 3.x y 4.x, viéndose reflejado en C

(b) ¿Cuáles son sus funciones principales?

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

(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 última versión es la **5.16** (9 de enero del 2022)

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?

Se pueden tener varios instalados (Versión anterior o algún kernel virtual), pero en realidad estaría funcionando solo uno, ya que es la parte de un sistema operativo que administra y controla los recursos y procesos

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

En /boot. El primer sector del disco se llama boot sector. Contiene información general de donde se almacena el Kernel y como se arranca

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

Si, es un núcleo monolítico (todo en un solo proceso) híbrido: Los drivers y código del Kernel se ejecutan en modo privilegiado con acceso irrestricto al hardware, aunque 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?

El Shell (intérprete de comandos) es el programa que recibe lo que se escribe en la terminal y lo convierte en instrucciones para el sistema operativo. Un intérprete de comandos es un programa que lee las entradas del usuario y las traduce a instrucciones que el sistema es capaz de entender y utilizar.

(b) ¿Cuáles son sus funciones?

Actua como interfaz para comunicar al usuario con el sistema operativo mediante una ventana que espera comandos textuales ingresados por el usuario en el teclado, los interpreta y los entrega al SO para su ejecución. La respuesta del SO es mostrada al usuario en la misma ventana. A continuación, la shell queda esperando más instrucciones. Se interactúa con la información de la manera más simple posible, sin gráficas, solo el texto.

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

Dentro de GNU/Linux y Unix, existen tres grandes familias de Shells, estas son: Korn-Shell (ksh), Bourne-Shell (sh) y C-Shell (csh). Estas se diferencian entre sí básicamente en la sintaxis de sus comandos y en la interacción con el usuario. El más usado hoy en dia es bash (-su nombre es un acrónimo de bourne-again shell, haciendo un juego de palabras sobre el Bourneshell-).

- Bourne-Shell (sh). Creado por S. Bourne, es el más utilizado en la actualidad. Su símbolo del sistema es $. Es el shell estándar y el que se monta en casi todos los sistemas UNIX/Linux.
- C-Shell (csh). Procedente del sistema BSD, proporciona funciones tales como control de trabajos, historial de órdenes, etc. Ofrece importantes características para los programadores que trabajan en lenguaje C. Su símbolo del sistema es %.
- Korn-Shell (ksh). Escrito por David Korn, amplía el shell del sistema añadiendo historial de órdenes, edición en línea de órdenes y características ampliadas de programación.
- Bourne Again Shell (bash). Fue creado para usarlo en el proyecto GNU. BASH, por lo tanto, es un shell o intérprete de comandos GNU que incorpora la mayoría de distribuciones de Linux. Es compatible con el shell sh. Además, incorpora algunas características útiles de ksh y csh, y otras propias como la edición de línea de comandos, tamaño ilimitado del historial de comandos, control de los trabajos y procesos, funciones y alias, cálculos aritméticos con números enteros, etc. Su símbolo del sistema es nombre_usuario@nombre_equipo.

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

El shell nos permite ejecutar:

Comandos externos, por ejemplo: ls, cat, mkdir, etc.

- son programas ajenos al shell
- cuando se lanzan inician un nuevo proceso
- se buscan en los directorios indicados en la variable PATH

Comandos internos (builtin commands), por ejemplo: cd, bg, alias, eval, exec, pwd, etc.

- se ejecutan en el mismo proceso del shell, sin lanzar un nuevo proceso

En bash: para saber si un comando es externo o interno usar el comando interno type:

$ type cd cd is a

shell builtin $

type cat cat is

/bin/cat

La diferencia fundamental es que los internos están incorporados a la consola y se pueden ejecutar directamente, mientras que para los externos hay que indicar la ruta hasta la ubicación del comando.

Para los comandos externos puede ser que no tengamos que indicar la ruta hasta la ubicación del mismo de forma explícita, si esta ruta está incluida en la variable de entorno PATH.

También debemos tener precaución en el caso de que el comando exista tanto de forma interna y externa, ya que las dos versiones del comando pueden dar resultados distintos, por lo que si queremos estar seguros de que estamos ejecutando la versión externa debemos indicar la ruta del comando (p.e. pwd ó /bin/pwd)

Los comandos internos son nativos del shell de linux que estemos usando (bash por ejemplo). Estos se suelen encontrar en el directorio /usr/binLos externos se encuentran en la variables $PATH, no son nativos del shell de linux,Ser capaz de editar tu path es una habilidad importante para todo principiantede Linux.

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

La shell no forma parte del kernel básico del SO; sino que la misma "dialoga" con el kernel.

No es muy difícil darse cuenta de por qué el shell no es parte del kernel. Como el shell se usa para interpretar las órdenes del usuario y ejecutarlas, si el mismo estuviese en el kernel tendría acceso a instrucciones propias que usa el SO para la gestión de los distintos dispositivos del hardware. Razón por la cual se abstrae al usuario del manejo de dispositivos hardware, dejándolo al kernel.

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

Si es posible definir un intérprete de comandos distintos para cada usuario ya que la shell es iniciada por un proceso denominado "login" en donde cada usuario tiene asignado una shell por defecto que se puede personalizar, la misma se inicia cada vez que un usuario comienza a trabajar en su estación de trabajo (es decir se "loguea" en una terminal). Dentro del contenido del archivo /etc/passwd, se puede ver cual es la shell que cada usuario tiene asignada por defecto. Las Shell son programables.

6. Sistema de Archivos (File System):

(a) ¿Qué es?

El Filesystem se traduce como "sistema de archivos"; y es la forma en que dentro de un SO se organizan y se administran los archivos. Esa administración comprende:

- Métodos de acceso: cómo se acceden los datos contenidos en el archivo.
- Manejo de archivos: cómo actúan los mecanismos para almacenar, referenciar, compartir y proteger los archivos.
- Manejo de la memoria secundaria: Cómo se administra el espacio para los archivos en memoria secundaria.
- Mecanismos de integridad: con qué métodos se garantiza la incorruptibilidad del archivo.

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

Linux soporta gran variedad de sistemas de ficheros, desde sistemas basados en discos, como pueden ser **ext2, ext3, ReiserFS, XFS, JFS, UFS, ISO9660, FAT, FAT32 o NTFS**, a sistemas de ficheros que sirven para comunicar equipos en la red de diferentes sistemas operativos, como **NFS** (utilizado para compartir recursos entre equipos Linux) o **SMB** (para compartir recursos entre máquinas Linux y Windows).

GNU/Linux soporta una gran cantidad de tipos de sistema de archivos: adfs, affs, autofs, coda, coherent, cramfs, devpts, efs, ext2, ext3, hfs, hpfs, iso9660, jfs, minix, msdos, ncpfs, nfs, ntfs, proc, qnx4, reiserfs, romfs, smbfs, sysv, tmpfs, udf, ufs, umsdos, vfat, xenix, xfs

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

Linux no permite ver por defecto el contenido de las particiones de Windows. Para poder hacerlo será necesario que montemos la partición NTFS o FAT en la que está Windows. Aunque en estos momentos existen distribuciones de GNU-Linux que pueden realizar operaciones de lectura y escritura sobre ellas.

El Kernel de unix tiene soporte para el manejo de particiones FAT,NTFS

(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 de archivos es una estructura jerárquica en forma de árbol invertido, donde el directorio principal (raíz) es el directorio "/", del que cuelga toda la estructura del sistema. Este sistema de archivos permite al usuario crear, borrar y acceder a los archivos sin necesidad de saber el lugar exacto en el que se encuentran. No existen unidades físicas, sino archivos que hacen referencia a ellas.

/bin: En donde se residen los comandos principales de Linux como ls o mv.

/boot: Aquí se encuentran los cargadores de inicio y los archivos de inicio del sistema.

/dev: En esta ruta se encuentran montados todos los dispositivos físicos como el USBs o el DVDs.

/etc: Contiene la configuración de los paquetes instalados.

/home: Los usuarios del sistema tendrán su carpeta personal para colocar todas sus carpetas adicionales con su nombre como se muestra a continuación: /home/likegeeks.

/lib: Aquí se guardan las librerías de los paquetes instalados ya que estas librerías son compartidas por todos los paquetes. A diferencia de Windows, puedes encontrar duplicados en diferentes carpetas.

/media: En esa ruta se encuentran los dispositivos externos como los DVDs ylos pendrives USB, desde aquí puedes acceder a sus archivos desde aquí.

/mnt: Aquí se montan otras cosas como localizaciones de red y algunas distribuciones que puedas tener montadas en un pendrive o DVD.

/opt: Algunos paquetes opcionales se encuentran aquí y esta ruta es administrada por el administrador de paquetes.

/proc: Debido a que todo en Linux es un archivo, esta es una carpeta que tiene los procesos ejecutándose en el sistema, y puedes acceder a ellos para obtener información acerca de los procesos actuales.

/root: La carpeta home para el usuario root.

/sbin: Como /bin, pero con archivos binarios solo para el usuario root.

/tmp: Contiene los archivos temporales.

/usr: Aquí es donde las utilidades y los archivos se comparten entre usuarios en Linux.

/var: Contiene registros del sistema y otros datos variables

Directorio	Descripción
/	Es la raíz del sistema de directorios. Aquí se monta la partición principal Linux EXT.
/etc	Contiene los archivos de configuración de la mayoría de los programas.
/home	Contiene los archivos personales de los usuarios.
/bin	Contiene comandos básicos y muchos programas.
/dev	Contiene archivos simbólicos que representan partes del hardware, tales como discos duros, memoria...
/mnt	Contiene subdirectorios donde se montan (se enlaza con) otras particiones de disco duro, CDROMs, etc.
/tmp	Ficheros temporales o de recursos de programas.
/usr	Programas y librerías instalados con la distribución
/usr/local	Programas y librerías instalados por el administrador
/sbin	Comandos administrativos
/lib	Librerías varias y módulos ("trozos") del kernel
/var	Datos varios como archivos de log (registro de actividad) de programas, bases de datos, contenidos del servidor web, copias de seguridad...
/proc	Información temporal sobre los procesos del sistema (explicaremos esto más en profundidad posteriormente).

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

7. Particiones:

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

Una partición de disco, en mantenimiento, es el nombre genérico que recibe cada división presente en una sola unidad física de almacenamiento de datos. Toda partición tiene su propio sistema de archivos (formato); generalmente, casi cualquier sistema operativo interpreta, utiliza y manipula cada partición como un disco físico independiente, a pesar de que dichas particiones estén en un solo disco físico.

Ventajas y deventajas:

-Es una buena práctica separar los datos del usuario de las aplicaciones y/o Sistema Operativo instalado

-Tener una partición de Restore de todo el sistema

-Poder ubicar el Kernel en una partición de solo lectura, o una que nisiquiera se monta (no está disponible a los usuarios).

-Particionar demasiado un disco puede tener desventajas. (Al achicar el tamaño del disco rígido físico, puede ocurrir que archivos grandes no entren en el tamaño de una de las particiones nuevas).

Partición primaria: Son las divisiones primarias del disco que dependen de una tabla de particiones, y son las que detecta el ordenador al arrancar, por lo que es en ellas donde se instalan los sistemas operativos. Puede haber un máximo de cuatro, y prácticamente cualquier sistema operativo las detectará y asignará una unidad siempre y cuando utilicen un sistema de

archivo compatible. Un disco duro completamente formateado contiene en realidad una partición primaria ocupando todo su espacio.

Partición extendida o secundaria: Fue ideada para poder tener más de cuatro particiones en un disco duro, aunque en ella no se puede instalar un sistema operativo. Esto quiere decir que sólo la podremos usar para almacenar datos. Sólo puede haber una de ellas, aunque dentro podremos hacer tantas otras particiones como queramos. Si utilizas esta partición, el disco sólo podrá tener tres primarias, siendo la extendida la que actúe como cuarta.

Partición lógica: Son las particiones que se hacen dentro de una partición extendida. Lo único que necesitarás es asignarle un tamaño, un tipo de sistema de archivos (FAT32, NTFS, ext2,...), y ya estará lista para ser utilizada. Funcionan como si fueran dispositivos independientes, y puedes utilizarla para almacenar cualquier archivo

“

Debido al tamaño acotado en el 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*
- *Una de las 4 particiones puede ser extendida, la cual se subdivide en volúmenes lógicos*
- *Partición primaria: división cruda del disco (puede haber 4 por disco). Se almacena información de la misma en el MBR*
- *Partición extendida: sirve para contener unidades lógicas en su interior. Solo puede existir una 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. Las particiones de este tipo se conectan como una lista enlazada*

“

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

- Disqueteras ○ Primera disquetera: **/dev/fd0** (en Windows sería la disquetera A:) ○ Segunda disquetera: **/dev/fd1**
- Discos duros (en general: **/dev/hdx#,** donde x es el disco y # es la partición)
  - ○ Primer disco duro: (todo el disco) **/dev/hda**
    - Particiones primarias
      - Primera partición primaria: **/dev/hda1**
      - Segunda partición primaria: **/dev/hda2**
      - Tercera partición primaria: **/dev/hda3**
      - Cuarta partición primaria: **/dev/hda4** ▪ Particiones lógicas
        - Primera partición lógica: **/dev/hda5**
        - Sucesivamente: **/dev/hda#**
  - ○ Segundo disco duro: (todo el disco**) /dev/hdb**
    - Particiones primarias

- Primera partición primaria: **/dev/hdb1**
- Segunda partición primaria: **/dev/hdb2**
- Tercera partición primaria: **/dev/hdb3**
- Cuarta partición primaria: **/dev/hdb4**

▪ Particiones lógicas

- Primera partición lógica: **/dev/hdb5**
- Sucesivamente: **/dev/hdb#**

• Discos SCSI ○ Primer disco SCSI: **/dev/sda** ○ Segundo disco SCSI: **/dev/sdb** ▪ Sucesivamente ...

Primer CD-ROM SCSI: /dev/scd0, también conocido como **/dev/sr0**

(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.

Como mínimo es necesario una partición (para el /). Es recomendable crear al menos 2 (/ y SWAP). Usualmente se suelen tener tres, una para el sistema/programas (/), otra para los datos (/home) y otra para swap.

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

- 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.

Existen 2 tipos:

- 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

partman

Herramienta original de Linux para particionar discos. Esta «navaja suiza» también puede ajustar el tamaño de las particiones, crear sistemas de ficheros (como se llama en Windows a “formatear”) y asignarlos a sus respectivos puntos de montaje.

fdisk

Es la herramienta original de Linux para particionar discos, buena para expertos.

Sea cuidadoso si tiene una partición de FreeBSD en su máquina. Los núcleos instalados traen soporte para este tipo de partición, pero la manera en que fdisk la representa, puede (o no) ser un poco diferente.

cfdisk

Una herramienta para particionar a pantalla completa, muy fácil de usar. Recomendada para la mayoría de los usuarios.

cfdisk no reconoce las particiones de FreeBSD, y nuevamente, los dispositivos mostrados en pantalla pueden ser un tanto diferentes a los que realmente tiene

8. Arranque (bootstrap) de un Sistema Operativo:

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

En las arquitecturas x86 la BIOS de la motherboardes un chip especial que guarda configuración inicial de la computadora. La BIOS es el sistema básico de entrada/salida (Basic Input-Output System) y ya viene incorporado a la placa base a través de la memoria flash. Es básicamente la encargada del manejo y configuración de la placa base y sus componentes.

Su función principal es la de iniciar los componentes de hardware y lanzar el sistema operativo de un ordenador cuando lo encendemos (a través del MBC). También carga las funciones de gestión de energía y temperatura del ordenador.

Cuando enciendes tu ordenador lo primero que se carga en él es el BIOS. Este firmware entonces se encarga de iniciar, configurar y comprobar que se encuentre en buen estado el hardware del ordenador, incluyendo la memoria RAM, los discos duros, la placa base o la tarjeta gráfica. Cuando termina selecciona el dispositivo de arranque (disco duro, CD, USB etcétera) y procede a iniciar el sistema operativo, y le cede a él el control de tu ordenador.

En otras arquitecturas también existe, pero se lo conoce conotro nombre:

- PoweronReset + IPL en mainframe
- OBP (OpenBoot PROM): en SPARC

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

EFI (Extensible Firmware Interface), UEFI en la práctica (UnifiedExtensible Firmware Interface), es una especificación que desarrolló Intel, quees un nexo entre el sistema operativo y el firmware. Desde este punto de vista puedeverse como una alternativa para reemplazar la BIOS

Es el firmware sucesor, escrito en C, del BIOS. A mediados de la década pasada las empresas tecnológicas se dieron cuenta de que el BIOS estaba quedándose obsoleto, y 140 de ellas se unieron en la fundación UEFI para renovarla y reemplazarla por un sistema más moderno.En

esencia, todo lo que hace el BIOS lo hace también la UEFI. Pero también tiene otras funciones adicionales y mejoras sustanciales, como una interfaz gráfica mucho más moderna, un sistema de inicio seguro, una mayor velocidad de arranque o el soporte para discos duros de más de 2 TB

**(un firmware es un software que maneja físicamente al hardware)*

(c) ¿Qué es el MBR? ¿Qué es el MBC?

El MBR (master boot record) es el primer sector del disco (cilindro 0, cabeza 0, sector 1). Esto se carga a memoria y se ejecuta. Es un registro de arranque principal que puede contener un código de arranque denominado MBC (master boot code) y una marca de 2 bytes que indica su presencia o puede solamente contener la tabla de particiones. En el último caso el BIOS ignora este MBR. Existe un MBR en todos los discos y 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 de sector: 512 bytes.

El MBC es un pequeño código que permite arrancar el SO. La última acción del BIOS es leer el MBC, lo lleva a memoria y lo ejecuta.

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

UEFIutiliza el sistema GPT (GUID partition table) 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, es parte de EFI y puede verse como una sustitución del MBR.

(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 (en inglés «bootloader») es un programa sencillo que no tiene la totalidad de las funcionalidades de un sistema operativo, y que está diseñado exclusivamente para preparar todo lo que necesita el sistema operativo para funcionar. Normalmente se utilizan los cargadores de arranque multietapas, en los que varios programas pequeños se suman los unos a los otros, hasta que el último de ellos carga el sistema operativo.

En los ordenadores modernos, el proceso de arranque comienza cuando la unidad central de procesamiento ejecuta los programas contenidos en una memoria de sólo lectura en una dirección predefinida y se configura la unidad central para ejecutar este programa, sin ayuda externa, al encender el ordenador.

Habitualmente se instala en el MBR y asume el rol de MBC. En una computadora en la que hay sólo un sistema operativo, no hay referencias a pantalla generalmente. Si hay un gestor de arranque, este programa nos permitirá elegir el sistema operativo a arrancar. El código del MBC de Windows, por ejemplo, busca en la tabla de particiones cual es la primer partición primaria con el flag de "booteo" activo y transfiere el control al código que se encuentra al comienzo de dicha partición: el PBR (partition boot record).

En el caso del sistema operativo Linux, se puede optar por distintos gestores de arranque, por ejemplo, LILO (Linux Loader), GRUB (Grand Unified Bootloader) o GAG (Gestor de arranque Gráfico). LILO no se basa en un sistema de archivos específico. Funciona en una variedad de sistemas de archivos. GRUB en cambio, debe comprender el sistema de archivos y el formato de los directorios2. GRUB tiene algunas ventajas con respecto a LILO: tiene una línea de

comandos interactiva como, permite arrancar desde una red, y podría considerarse más seguro

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

El BIOS (Basic I/O System) es el responsable de iniciar la carga del SO a través del MBC. Este 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 y 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.

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 se suceden en el proceso de parada (shutdown) de GNU/Linux?

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.

Si, ya que un disco rígido puede particionarse y en cada partición tener un sistema de archivos distinto (partición primaria), sería como tener varios discos distintos, uno con cada SO, por lo tanto, se necesitará un gestor de arranque como los descriptos arriba.

9. Archivos:

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

La base del sistema de archivos de Linux, es obviamente el archivo, que no es otra cosa que la estructura empleada por el sistema operativo para almacenar información en un dispositivo físico como un disco duro, un disquete, un CD-ROM o un DVD. Como es natural un archivo

puede contener cualquier tipo de información, desde una imagen en formato PNG o JPEG a un texto o una página WEB en formato HTML.

El sistema de archivos es la estructura que permite que Linux maneje los archivos que contiene. Todos los archivos de Linux tienen un nombre, el cual debe cumplir unas ciertas reglas:

- Un nombre de archivo puede tener entre 1 y 255 caracteres.
- Se puede utilizar cualquier carácter excepto la barra inclinada / y no es recomendable emplear los caracteres con significado especial en Linux, que son los siguientes: = ^ ~ ' " ` * ; - ? [ ] ( ) ! & ~ < >. Para emplear ficheros con estos caracteres o espacios hay que introducir el nombre del fichero entre comillas.
- Se pueden utilizar números exclusivamente si así se desea.
- Las letras mayúsculas y minúsculas se consideran diferentes, y por lo tanto no es lo mismo carta.txt que Carta.txt ó carta.Txt.

Como en Windows, se puede emplear un cierto criterio de "tipo" para marcar las distintas clases de ficheros empleando una serie de caracteres al final del nombre que indiquen el tipo de fichero del que se trata. Así, los ficheros de texto, HTML, las imágenes PNG o JPEG tienen extensiones .txt, .htm (o .html), .png y .jpg (o .jpeg) respectivamente. Pese a esto Linux sólo distingue tres tipos de archivos:

- Archivos o ficheros ordinarios, son los mencionados anteriormente.
- Directorios (o carpetas), es un archivo especial que agrupa otros ficheros de una forma estructurada.
- Archivos especiales, son la base sobre la que se asienta Linux, puesto que representan los dispositivos conectados a un ordenador, como puede ser una impresora. De esta forma introducir información en ese archivo equivale a enviar información a la impresora. Para el usuario estos dispositivos tienen el mismo aspecto y uso que los archivos ordinarios

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

VI

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). Puede usarse en cualquier tipo de terminal con un mínimo de teclas, lo cual lo hace difícil de usar al enfrentarse por primera vez al mismo.

MODOS DE VI:

Existen tres modos o estados de vi:

- Modo comando: este es el modo en el que se encuentra el editor cada vez que se inicia. Las teclas ejecutan acciones (comandos) que permiten mover el cursor, ejecutar comandos de edición de texto, salir de vi, guardar cambios, etc.
- Modo inserción o texto: este es el modo que se usa para insertar el texto. Existen varios comandos que se pueden utilizar para ingresar a este modo.

• Modo línea o ex: se escriben comandos en la última línea al final de la pantalla.

INICIO DE VI:

vi

Abre la ventana de edición sin abrir ningún archivo.

vi *archivo1*

Edita el archivo archivo1 si ya existe, de lo contrario, lo crea. Evidentemente se debe indicar el camino (path) que conduce al archivo (si existe) o el camino que conduce al directorio donde se desea crear el archivo (si este no existe).

MODO COMANDO:

El editor vi, como todo UNIX, diferencia mayúsculas de minúsculas. A continuación, se comentan algunos comandos útiles en el manejo del editor.

Movimiento del cursor:

Comando (teclas)	Acción
`Flechas`	Mover en la dirección de la flecha
`h`	Mover hacia la izquierda
`l`	Mover hacia la derecha
`k`	Mover hacia arriba
`j`	Mover hacia abajo
`1G`	Lleva el cursor hasta el comienzo del archivo
`G`	Lleva el cursor hasta el final del archivo

Cambio de modo comando a texto:

Comando	Acción
`i`	Inserta texto a la izquierda del cursor
`a`	Inserta texto a la derecha del cursor
`A`	Inserta texto al final de la línea donde se encuentra el cursor
`I`	Inserta texto al comienzo de la línea donde se encuentra el cursor
`o`	Abre una línea debajo de la actual
`O`	Abre una línea encima de la actual

Borrar texto:

Comando	Acción
x	Borra el carácter bajo el cursor
dd	Borra la línea donde se encuentra el cursor
ndd	Borra las próximas n líneas
D	Borra desde donde se encuentra el cursor hasta el final de la línea
dw	Borra desde donde se encuentra el cursor hasta el final de una palabra

Es importante destacar que todo lo que se borra queda almacenado en un buffer (área temporal de memoria), de modo que si se borró algo por error, puede volver a escribirse (si se hace antes de realizar otros cambios, es decir, inmediatamente luego de eliminar el texto por error. Esto se hace simplemente ejecutando el comando **p.**

Cortar y pegar:

Esto implica mover partes del archivo de un lugar a otro del mismo. Para esto se debe:

- Cortar el texto que se desea mover utilizando alguno de los comandos usados para borrar texto.
- Mover el cursor (con alguno de los comandos utilizados para desplazar el cursor en el texto) hasta el lugar donde se desee pegar el texto.
- Pegar el texto con el comando p.

Copiar y pegar:

Esta operación difiere de la anterior. En este caso lo que se hace es repetir partes del texto en otro lugar del archivo. Para esto se debe:

- Utilizar el comando yy, cuya función es copiar la línea donde se encuentra situado el cursor.
- Mover el cursor (con alguno de los comandos utilizados para desplazar el cursor en el texto) hasta el lugar donde se desee pegar el texto.
- Pegar el texto con el comando p.

Deshacer cambios:

Se puede deshacer el último cambio realizado, utilizando el comando u.

MODO TEXTO:

En este modo se ingresa el texto deseado. Para pasar de modo texto a modo comando simplemente se debe apretar la tecla ESC.

MODO LÍNEA:

Para ingresar al modo línea desde el modo comando, se debe utilizar alguna de las Siguientes teclas:

/

?

:

Para volver al modo comando desde el modo última línea, se debe apretar la tecla ENTER (al finalizar el comando) o la tecla ESC (que interrumpe el comando).

Buscar texto:

Comando	Acción
`/texto`	Busca hacia adelante la cadena de caracteres "texto"
`?texto`	Busca hacia atrás la cadena de caracteres "texto"

Salir de vi, salvar, no salvar cambios, etc.:

Comando	Acción
`:q`	Salir si no hubo cambios
`:q!`	Salir sin guardar cambios
`:w`	Guardar cambios
`:w archivo1`	Guardar cambios en archivo1
`:wq`	Guardar cambios y salir

MCEDIT

Midnight Commander es una aplicación que funciona en modo texto. La pantalla principal consiste en dos paneles en los cuales se muestra el sistema de ficheros. Se usa de un modo similar a otras aplicaciones que corren en el shell o interfaz de comandos de Unix. Las teclas de cursor permiten desplazarse a través de los ficheros, la tecla insertar se usa para seleccionar ficheros y las Teclas de función realizan tareas tales como borrar, renombrar, editar, copiar ficheros, etc. Las versiones más recientes de Midnight Commander incluyen soporte para el ratón para facilitar el manejo de la aplicación.

Incluye un editor llamado mcedit. Mcedit es un ejecutable independiente, el cual también puede ser usado de forma independiente a Midnight Commander. Esta aplicación permite visualizar el contenido de ficheros y disfrutar de características como la de resaltar la sintaxis para ficheros de código fuente de ciertos lenguajes de programación, y la capacidad de trabajar tanto en modo ASCII como en modo Hexadecimal. Los usuarios pueden reemplazar mcedit por el editor que prefieran.

Cat: El comando 'cat' imprimirá por pantalla el contenido del fichero sin ningún tipo de paginación ni posibilidad de modificarlo. Básicamente concatena archivos o la salida estándar en la salida estándar. Podemos pasarle parámetros como

More: Al igual que 'cat', 'more' permite visualizar por pantalla el contenido de un fichero de texto, con la diferencia con el anterior de que 'more' pagina los resultados. Primero mostrará por pantalla todo lo que se pueda visualizar sin hacer scroll y después, pulsando la tecla espacio avanzará de igual modo por el fichero.

Less: El comando 'less' es el más completo de los tres, pues puede hacer todo lo que hace 'more' añadiendo mayor capacidad de navegación por el fichero (avanzar y retroceder)

además de que sus comandos están basados en el editor 'vi', del cual se diferencia en que no tiene que leer todo el contenido del fichero antes de ser abierto. Tiene una gran cantidad de opciones y parámetros, como siempre lo recomendable:

(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.

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

El comando file dice la línea en la que esta posicionado el cursor en el editor de texto

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 carpeta ISO2017 (b)

Acceda a la carpeta (cd)

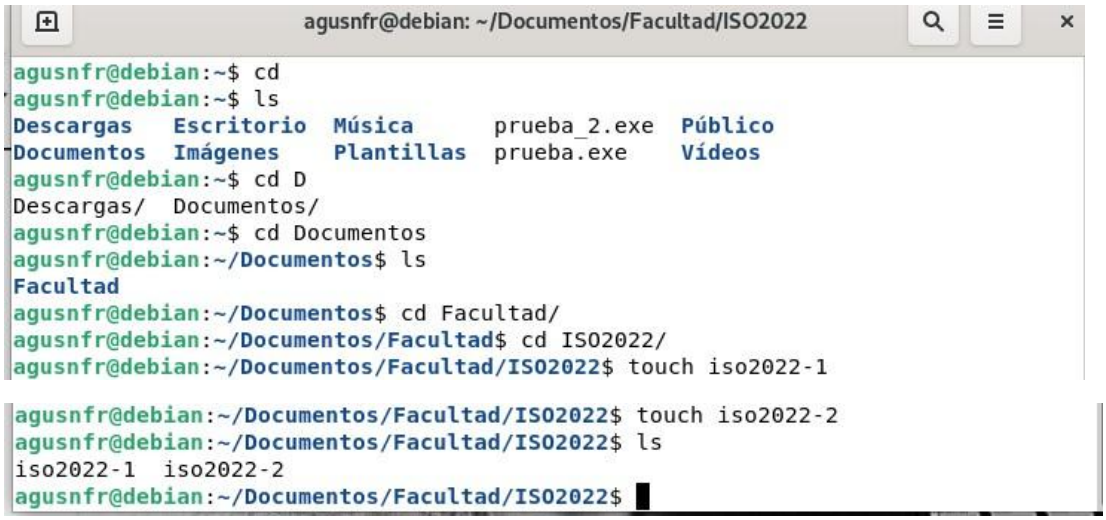
cd: cambio de directorio

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

crear cualquier tipo nuevo de archivo en sistemas Linux

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

directorio que desee, archivos y otros directorios anidados



(e) Visualizar la ruta donde estoy situado (pwd) pwd: se usa para localizar la ruta del directorio de trabajo en el que te encuentras **(f) Busque todos los archivos en los que su nombre contiene la cadena "iso*" (find)**

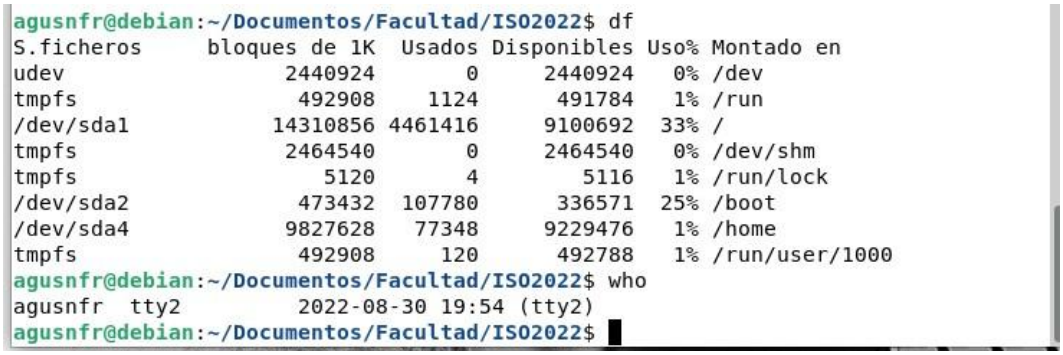
find: buscar archivos y directorios según sus permisos, tipo, fecha, propiedad, tamaño, etc.. También se puede combinar con otras herramientas como grep o sed.

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

df : Se usa para mostrar la cantidad en porcentaje y KB de espacio libre en disco disponible en Linux.

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

who: nos da información de los usuarios que están conectados al sistema y también otras informaciones como cuándo se arrancó el sistema y cuál es el nivel de ejecución del sistema.



(i) Acceder a el archivo iso2017-1 e ingresar Nombre y Apellido (j) Mostrar en pantalla las últimas líneas de un archivo (tail).

tail: se utiliza para mostrar las últimas líneas de un archivo (de texto) o para restringir la salida de un comando de Linux a un ámbito concreto

11. Investigue su funcionamiento y parámetros más importantes:

(a) shutdown

El comando shutdown se utiliza para apagar o reiniciar Linux desde la terminal.

Detener el sistema de forma segura. shutdown

[OPTIONS] [TIME] [MESSAGE]

(b) reboot

Reinicia SO

-reboot, -r: reinicia el sistema

(c) halt

El comando halt se utiliza para apagar el ordenador

halt [-d | -f | -h | -n | -i | -p | -w]

reboot [-d | -f | -i | -n | -w]

poweroff [-d | -f | -h | -n | -i | -w]

OPCIONES:

- d	**No escribir registro wtmp (en el archivo /var/log/wtmp) El flag -n implica -d**
- h	Poner todos los discos duros del sistema en modo de espera antes de que el sistema se detenga o apague
- n	No sincronizar antes de reiniciar o detener
- i	Apagar todas las interfaces de red.
- p	Cuando detenga el sistema, lo apaga también. Esto es por defecto cuando el halt se llama como poweroff.
- w	No reiniciar o detener, sólo escribir el registro wtmp (en el archivo /var/log/wtmp)

(d) locate

El comando locate es una alternativa útil, ya que es más rápido que find para realizar búsquedas. Eso se debe a que sólo escanea tu base de datos de Linux en lugar de todo el sistema locate [my-file]

(e) uname

Si utilizamos el comando sin argumentos nos entregara la palabra Linux extraída de la información del Kernel.

$ uname

Si deseamos extraer la información de la versión del kernel utilizamos el parámetro -r.

$ uname -r

Si deseamos extraer la fecha de cuando la versión del kernel fue liberada utilizamos el parámetro -v. $ uname -v

(f) dmesg

Se usa para examinar o controlar el ring buffer del kernel. La acción predeterminada es mostrar todos los mensajes del ring buffer.

Debido a toda la información desplegada, es difícil llevar a cabo alguna tarea de administración allí. Podemos hacer uso del parámetro "-H" con el fin de indicarle a dmesg que la salida sea legible para los usuarios, lo cual simplificará las tareas de soporte. Allí encontramos detalles mucho más claros sobre el anillo del kernel.

Otra alternativa para realizar un análisis con dmesg es con el parámetro "-w", el cual nos permite escribir un script para analizar el resultado usando una expresión regular con el fin de filtrar los eventos para su posterior análisis:

(g) lspci

lspci es un comando para los sistemas operativos Unix-like que imprime listas con información detallada sobre todos los Buses y dispositivos del sistema **(h) at**

Permite programar tareas únicas en nuestro sistemas GNU/Linux.

N os permite programar tareas para que se ejecuten a determinada fecha y hora .

at [hora] [fecha]

El comando at, nos puede ser útil para apagar el sistema a una hora especifica, realizar una copia de seguridad única, enviar un correo electrónico como recordatorio a la hora especificada, entre otras muchas cosas.

(i) netstat

netstat (estadísticas de red) es una herramienta de línea de comandos que muestra las conexiones de red (entrantes y salientes), tablas de enrutamiento y una serie de estadísticas de interfaz de red.

(j) mount

El mandato mount ordena al sistema operativo que haga que un sistema de archivos esté disponible para su utilización en una ubicación determinada (el punto de montaje).

(k) umount

Este comando permite desmontar un sistema de archivos montado previamente.El uso del comando umount garantiza que toda la información mantenida en memoria por el sistema operativo se escriba en el dispositivo antes de desmontarlo

Sintaxis: umount dispositivo | punto_montaje

(l) head

El comando head muestra de modo predeterminado las diez primeras líneas de un archivo. Se puede modificar esta opción a las N primeras líneas del archivo con la sintaxis head -nN.

(m) losetup

se usa para asociar loop devices con archivos regulares o block devices, también para desacoplar loop devices, y para hacer queries del status de un loop device. (dispositivo iterador = loop device)

(n) write

El comando write permite mandar un mensaje a otro usuario del sistema especificando como parámetros el usuario al que enviar el mensaje y la TTY asociada:

$ write usuario tty.

Para finalizar la escritura del mensaje y enviarlo presionamos CTRL + D. La TTY se especificará cuando el usuario al que enviemos el mensaje tenga más de una sesión abierta. En el caso de que no le especificamos, se enviará automáticamente a la tty del usuario con actividad más reciente **(ñ) mkfs**

se utiliza para dar formato a un dispositivo de almacenamiento de bloque con un determinado sistema de archivos. **(o) fdisk (con cuidado)**

es una utilidad de línea de comandos basada en texto para ver y administrar particiones de disco duro en Linux. Con fdisk puedes ver, crear, cambiar el tamaño, eliminar, cambiar, copiar y mover particiones.más info de Fdisk: https://maslinux.es/comando-fdisk-para-administrarparticiones-de-disco-en-gnulinux/

12. Investigue su funcionamiento y parámetros más importantes:

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

/bin: Aquí están los comandos que pueden usar todos los usuarios (incluido el root).

13. Editor de textos:

(a) Nombre al menos 3 editores de texto que puede utilizar desde la línea de comandos. vi, vim, neovim (nvim), nano y emacs.

(b) ¿En qué se diferencia un editor de texto de los comandos cat, more o less? Enumere los modos de operación que posee el editor de textos vi.

Los editores de texto permiten editar los archivos de texto. cat, more, less solo permiten visualizarlos.

vi tiene tres modos:

- insert mode: este modo se usa para la edición de texto normal. El modo reemplazar es una variación del modo insertar que reemplaza texto en lugar de insertarlo.
- command mode: este modo se usa para la exploración de archivos, copiar y pegar, y ejecutar comandos simples. Con este modo, también se realizan funciones como deshacer, rehacer y otras.

- ex mode: este modo se usa para guardar, cerrar y abrir archivos, así como también para buscar y reemplazar, y otras operaciones más complejas. Desde este modo, es posible insertar el resultado de programas en el archivo actual, configurar vim, etc. Todo lo que es posible usando ex se puede hacer desde este modo.

(c) Nombre los comandos más comunes que se le pueden enviar al editor de textos vi.

En ex mode:

Comando	Resultado
`:wq`	Guarda y cierra el archivo actual.
`:x`	Guarda el archivo actual si hay cambios sin guardar, y luego lo cierra.
`:w`	Guarda el archivo actual y permanece en el editor.
`:w <filename>`	Guarda el archivo actual bajo un nombre de archivo diferente.
`:q`	Cierra el archivo actual (solo si no hay cambios sin guardar).
`:q!`	Cierra el archivo actual, e ignora los cambios no guardados.

En command mode:

Tecla	Resultado
`i`	Cambia al modo *insertar* y comienza a insertar *antes* de la posición actual del cursor (insertar).
`a`	Cambia al modo *insertar* y comienza a insertar *luego* de la posición actual del cursor (anexar).
`I`	Mueve el cursor hasta el *inicio* de la línea actual y cambia al modo *insertar*.
`A`	Mueve el cursor hasta el *final* de la línea actual y cambia al modo *insertar*.
`R`	Cambia al modo *replace*, y comienza en el carácter bajo el cursor. En el modo *replace*, no se inserta texto, sino que cada carácter que ingresa reemplaza a un carácter del documento actual. (**`vim`** y **`vi`** también vienen con comandos de reemplazo más potentes; estos se analizan en otra sección.)
`o`	Abra una nueva línea *debajo* de la actual y cambie inmediatamente al modo *insertar*.
`O`	Abra una nueva línea *arriba* de la actual y cambie al modo *insertar*.

13. Proceso de Arranque SystemV (https://github.com/systeminit/si):

(a) Enumere los pasos del proceso de inicio de un sistema GNU/Linux, desde que se prende la PC hasta que se logra obtener el login en el sistema.

1. Se empieza a ejecutar el código del BIOS
2. El BIOS ejecuta el POST
3. El BIOS lee el sector de arranque (MBR)
4. Se carga el gestor de arranque (MBC)
5. El bootloader carga el kernel y el initrd
6. Se monta el initrd como sistema de archivos raíz y se inicializan componentes esenciales (ej.:
   scheduler)
7. El Kernel ejecuta el proceso init y se desmonta el initrd
8. Se lee el /etc/inittab
9. Se ejecutan los scripts apuntados por el runlevel 1
10. El final del runlevel 1 le indica que vaya al runlevel por defecto

11. Se ejecutan los scripts apuntados por el runlevel por defecto
12. El sistema está listo para usarse

(b) Proceso INIT. ¿Quién lo ejecuta? ¿Cuál es su objetivo?

El proceso INIT es ejecutado por el kernel, una vez que este ha sido cargado en memoria, y su función es cargar todos los subprocesos necesarios para el correcto funcionamiento del SO. Es el encargado de montar los filesystems y de hacer disponible los demás dispositivos.

El proceso init posee el PID 1 y se encuentra en /sbin/init (en SysV se lo configura a través del archivo /etc/inittab). No tiene padre y es el padre de todos los procesos (pstree).

(c) RunLevels. ¿Qué son? ¿Cuál es su objetivo?

Los RunLevels son los modos en que arranca Linux, y cada uno de estos es responsable de levantar (iniciar) o bajar (parar) una serie de servicios.

(d) ¿A qué hace referencia cada nivel de ejecución según el estándar? ¿Dónde se define qué Runlevel ejecutar al iniciar el sistema operativo? ¿Todas las distribuciones respetan estos estándares?

Según el estándar:

- 0: halt (parada)
- 1: single user mode (monousuario)
- 2: multiuser, without NFS (modo multiusuario sin soporte de red)
- 3: full multiuser mode console (modo multiusuario completo por consola)
- 4: no se utiliza
- 5: X11 (modo multiusuario completo con login gráfico basado en X)
- 6: reboot

En sistemas que utilizan el esquema SysVinit, el nivel de ejecución por defecto se especifica en el archivo /etc/inittab. No todas las distribuciones siguen estos estándares de runlevel de manera estricta. En los sistemas modernos, muchas distribuciones han reemplazado SysVinit con systemd, que utiliza "targets" en lugar de runlevels para definir los estados del sistema.

(e) Archivo /etc/inittab. ¿Cuál es su finalidad? ¿Qué tipo de información se almacena en él? ¿Cuál es la estructura de la información que en él se almacena?

El archivo /etc/inittab es un archivo de configuración cuyo como propósito principal es definir el nivel de ejecución (runlevel) por defecto al que el sistema debe arrancar, y especificar cómo se deben gestionar ciertos procesos del sistema durante el arranque y apagado.

Entonces:

- Define el runlevel por defecto que el sistema debe usar al iniciar.
- Especifica qué procesos se deben iniciar, gestionar y reiniciar en ciertos niveles de ejecución.
- Configura el comportamiento del sistema cuando cambian los runlevels.

El archivo contiene una lista de procesos que deben ejecutarse para diferentes runlevels y eventos. Cada línea en el archivo define un proceso o acción en el sistema.

Su estructura es: id:nivelesEjecución:acción:proceso

- Id: identifica la entrada en inittab (1 a 4 caracteres)
- NivelesEjecución: el/los niveles de ejecución en los que se realiza la acción • Acción: describe la acción a realizar: ○ wait: inicia cuando entra al runlevel e init espera a que termine ○ initdefault
  ○ ctrlaltdel: se ejecutará cuando init reciba la señal SIGINT ○ off, respawn, once, sysinit, boot, bootwait, powerwait, etc.
- Proceso: el proceso exacto que será ejecutado

(f) Suponga que se encuentra en el runlevel <X>. Indique qué comando(s) ejecutaría para cambiar al runlevel <Y>. ¿Este cambio es permanente? ¿Por qué?

Para cambiar de un runlevel a otro se utiliza el comando init N, donde N es el número de runlevel al que se desea cambiar: *$ init <Y>*

Este cambio es temporal y solo afecta al estado actual del sistema. El cambio no es permanente porque el sistema volverá a su runlevel predeterminado en el próximo reinicio. Para hacer un cambio permanente en el runlevel predeterminado, se debe modificar el archivo de configuración /etc/inittab.

(g) Scripts RC. ¿Cuál es su finalidad? ¿Dónde se almacenan? Cuando un sistema GNU/Linux arranca o se detiene se ejecutan scripts, indique cómo determina qué script ejecutar ante cada acción. ¿Existe un orden para llamarlos? Justifique.

Cuando init entra en un runlevel, llama al script rc con un argumento numérico especificando el nivel de ejecución al que ir. Entonces, el script rc inicia y detiene servicios en el sistema según sea necesario para llevar el sistema a ese nivel de ejecución.

Todos los scripts RC se encuentran en el directorio /etc/rc.d/, que contiene un subdirectorio para cada nivel de ejecución (rc0.d, …, rc6.d). Dentro de cada uno de estos subdirectorios hay enlaces simbólicos a los scripts maestros almacenados en /etc/rc.d/init.d/.

Los enlaces simbólicos se nombran con el formato: [S|K]<orden><nombreScript>.

Los archivos que comienzan con S mayúscula representan scripts que se inician al entrar en ese nivel de ejcución, mientras que los archivos que comienzan con una K mayúscula representan scripts que se detienen. Los números especifican el orden en que deben ejecutarse los scripts.

Por ejemplo, un demonio puede tener un script llamado S35daemon en rc3.d/, y un script llamado K65daemon para detenerlo en rc2.d/. Tener los números al principio del nombre del archivo hace que se ordenen, y se procesen, en el orden deseado.

14. SystemD(https://github.com/systemd/systemd):

(a) ¿Qué es sytemd?

systemd es un sistema que centraliza la administración de daemons y librerias del sistema. El daemon systemd reemplaza al proceso init (systemd pasa a tener el PID 1).

systemd mejora el paralelismo de booteo y es compatible con SystemV, si es llamado como init, los runlevels son reemplazados por targets, y al igual que con upstart, el archivo /etc/inittab no existe más.

(b) ¿A qué hace referencia el concepto de Unit en SystemD?

Las unidades de trabajo son denominadas units de tipo:

- Service: controla un servicio particular (.service).
- Socket: encapsula IPC, un sockect del sistema o file system FIFO (.socket) → sockect-based activation.
- Target: agrupa units o establece puntos de sincronización durante el booteo (.target) → dependencia de unidades
- Snapshot: almacena el estado de un conjunto de unidades que puede ser restablecido más tarde (.snapshot), etc.

Estas units pueden tener dos estados: active o inactive.

(c) ¿Para qué sirve el comando systemctl en SystemD?

- Administrar servicios: se puede iniciar, detener, reiniciar, habilitar y deshabilitar servicios del sistema.
- Ver el estado de un servicio: mostrar si un servicio está activo o inactivo, proporcionado información de errores o fallos.
- Cambiar el estado del sistema: sería como cambiar el runlevel.
- Gestionar el arranque y apagado del sistema.

(d) ¿A qué hace referencia el concepto de target en SystemD?

Un target en systemd es una forma flexible de agrupar unidades y definir el estado del sistema. Reemplazan los runlevels tradicionales de SysVinit y permiten a los administradores configurar con precisión qué servicios o unidades se deben iniciar, detener o reiniciar cuando se cambia el estado del sistema.

(e) Ejecute el comando pstree. ¿Qué es lo que se puede observar a partir de la ejecución de este comando?

pstree muestra los procesos en ejecución en forma de árbol. El árbol tiene su raíz en init o systemd, depende del sistema de inicio que utilice el sistema.

15. Usuarios:

(a) ¿Qué archivos son utilizados en un sistema GNU/Linux para guardar la información de los usuarios?

El archivo /etc/passwd se utiliza para almacenar información sobre los usuarios locales y para cada usuario hay siete campos separados por dos puntos:
username:password:uid:gid:GECOS:/home/dir:shell

1. username es el nombre del usuario (login).
2. password es donde se guardaban las contraseñas en formato cifrado tradicionalmente. Actualmente, se guardan cifradas en un archivo aparte con el nombre /etc/shadow (esto se indica colocando una x en el campo).

3. UID es un ID único de usuario, un número que identifica al usuario en el nivel más básico de forma unívoca.

4. GID es el número de ID de grupo principal del usuario.

5. GECOS es un texto arbitrario que, por lo general, incluye el nombre real del usuario y otra información adicional (mail, teléfono, etc.).

6. /home/dir es la ubicación donde se encuentran los datos personales del usuario y los archivos de configuración.

7. shell es la shell por defecto de los procesos y usuarios. La shell /sbin/nologin se utiliza para bloquear el inicio de sesión en el sistema de forma interactiva y es muy común utilizarla en cuentas de usuarios que representan procesos o servicios en lugar de usuarios humanos.

(b) ¿A qué hacen referencia las siglas UID y GID? ¿Pueden coexistir UIDs iguales en un sistema GNU/Linux? Justifique.

- UID (User ID): es un identificador único de usuario. Cada usuario tiene su propio UID, que es un número que lo identifica de manera unívoca. Por convención, muchas distribuciones de GNU/Linux asignan por defecto el UID 1000 al primer usuario creado en el sistema y luego asignan a los usuarios nuevos el primer número de UID disponible en el rango, a partir de la UID 1000 en adelante (a menos que se especifique uno explícitamente).
- GID (Group ID): es un identificador único de grupo. Cada grupo de usuarios tiene su propio GID, para identificarlo de manera unívoca.
    - Los grupos locales están definidos en /etc/group.
    - Cada usuario tiene exactamente un grupo principal y pueden ser miembros de ninguno o más grupos adicionales.
    - Para los usuarios locales, el grupo principal está definido por el número de GID del grupo indicado en el cuarto campo de /etc/passwd.
    - Generalmente, el grupo principal es propietario de los nuevos archivos creados por el usuario.
    - Normalmente, el grupo principal de un usuario creado recientemente es un grupo creado con el mismo nombre que el del usuario. El usuario es el único miembro de este grupo privado de usuarios (UPG).

(c) ¿Qué es el usuario root? ¿Puede existir más de un usuario con este perfil en GNU/Linux? ¿Cuál es la UID del root?

root es un superusuario, un usuario que tiene todo el poder sobre el sistema. Este usuario tiene el poder de anular los privilegios normales del sistema de archivos y se utiliza para manejar y administrar el sistema. UID 0 siempre se asigna a la cuenta de root.

Es necesario contar con privilegios de superusuario (root) para poder realizar tareas, como la instalación o eliminación de software, y para administrar los directorios y los archivos del sistema. La mayoría de los dispositivos solo pueden ser controlados por el usuario root, pero existen algunas excepciones (por ejemplo, los dispositivos desmontables, como los dispositivos USB). El usuario root tiene poder ilimitado para dañar el sistema: eliminar archivos y directorios, eliminar cuentas de usuarios, agregar puertas traseras, etc.

(d) Agregue un nuevo usuario llamado iso2017 a su instalación de GNU/Linux, especifique que su home sea creada en /home/iso_2017, y hágalo miembro del grupo catedra (si no existe, deberá crearlo). Luego, sin iniciar sesión como este usuario cree un archivo en su home personal que le pertenezca. Luego de todo esto, borre el usuario y verifique que no queden registros de él en los archivos de información de los usuarios y grupos.

```
root@c04685c6d964:/ISO# groupadd catedra
root@c04685c6d964:/ISO# sudo useradd -d "/home/iso/2017" -m iso2017
root@c04685c6d964:/ISO# sudo usermod -a -G catedra iso2017
root@c04685c6d964:/ISO#
root@c04685c6d964:/ISO# cat /etc/passwd | grep "iso2017"
Dockerfile          Practica-1/         docker-compose.yaml
root@c04685c6d964:/ISO# cat /etc/passwd | grep "iso2017" && cat /etc/group | grep "catedra"
iso2017:x:1000:1001::/home/iso/2017:/bin/sh
catedra:x:1000:iso2017
root@c04685c6d964:/ISO# cd /home/iso/2017:/bin/sh
bash: cd: /home/iso/2017:/bin/sh: No such file or directory
root@c04685c6d964:/ISO# cd /home/iso/2017:/bin/sh
bash: cd: /home/iso/2017:/bin/sh: No such file or directory
root@c04685c6d964:/ISO# cd /home/iso_2017
bash: cd: /home/iso_2017: No such file or directory
root@c04685c6d964:/ISO# cd /home/iso2017
bash: cd: /home/iso2017: No such file or directory
root@c04685c6d964:/ISO# cd
root@c04685c6d964:~#
What's next:
    Try Docker Debug for seamless, persistent debugging tools in any container or image → docker debug c04685c6d964
    Learn more at https://docs.docker.com/go/debug-cli/
PS C:\Users\matia\OneDrive\Escritorio\Matias\Facultad\4to Semestre\ISO\Practicas\Resoluciones> docker exec -it c04685c6d964 bash
root@c04685c6d964:/ISO# cd /home/iso/2017
root@c04685c6d964:/home/iso/2017# touch ejemplo.txt
root@c04685c6d964:/home/iso/2017# cat /etc/passwd | grep "iso2017"
iso2017:x:1000:1001::/home/iso/2017:/bin/sh
root@c04685c6d964:/home/iso/2017# userdel iso2017
root@c04685c6d964:/home/iso/2017# cat /etc/passwd | grep "iso2017"
root@c04685c6d964:/home/iso/2017#
```

(e) Investigue la funcionalidad y parámetros de los siguientes comandos:

- useradd ó adduser:

```
leo@leo: $ useradd --help
Usage: useradd [options] LOGIN
       useradd -D
       useradd -D [options]

Options:
      --badname                 do not check for bad names
  -b, --base-dir BASE_DIR       base directory for the home directory of the
                                new account
      --btrfs-subvolume-home    use BTRFS subvolume for home directory
  -c, --comment COMMENT         GECOS field of the new account
  -d, --home-dir HOME_DIR       home directory of the new account
  -D, --defaults                print or change default useradd configuration
  -e, --expiredate EXPIRE_DATE  expiration date of the new account
  -f, --inactive INACTIVE       password inactivity period of the new account
  -F, --add-subids-for-system   add entries to sub[ud]id even when adding a system user
  -g, --gid GROUP               name or ID of the primary group of the new
                                account
  -G, --groups GROUPS           list of supplementary groups of the new
                                account
  -h, --help                    display this help message and exit
  -k, --skel SKEL_DIR           use this alternative skeleton directory
  -K, --key KEY=VALUE           override /etc/login.defs defaults
  -l, --no-log-init             do not add the user to the lastlog and
                                faillog databases
  -m, --create-home             create the user's home directory
  -M, --no-create-home          do not create the user's home directory
  -N, --no-user-group           do not create a group with the same name as
                                the user
  -o, --non-unique              allow to create users with duplicate
                                (non-unique) UID
  -p, --password PASSWORD       encrypted password of the new account
  -r, --system                  create a system account
  -R, --root CHROOT_DIR         directory to chroot into
  -P, --prefix PREFIX_DIR       prefix directory where are located the /etc/* files
  -s, --shell SHELL             login shell of the new account
  -u, --uid UID                 user ID of the new account
  -U, --user-group              create a group with the same name as the user
  -Z, --selinux-user SEUSER     use a specific SEUSER for the SELinux user mapping
      --extrausers              Use the extra users database
```

- usermod:

```
leo@leo:~$ usermod --help
Usage: usermod [options] LOGIN

Options:
  -a, --append                  append the user to the supplemental GROUPS
                                mentioned by the -G option without removing
                                the user from other groups
  -b, --badname                 allow bad names
  -c, --comment COMMENT         new value of the GECOS field
  -d, --home HOME_DIR           new home directory for the user account
  -e, --expiredate EXPIRE_DATE  set account expiration date to EXPIRE_DATE
  -f, --inactive INACTIVE       set password inactive after expiration
                                to INACTIVE
  -g, --gid GROUP               force use GROUP as new primary group
  -G, --groups GROUPS           new list of supplementary GROUPS
  -h, --help                    display this help message and exit
  -l, --login NEW_LOGIN         new value of the login name
  -L, --lock                    lock the user account
  -m, --move-home               move contents of the home directory to the
                                new location (use only with -d)
  -o, --non-unique              allow using duplicate (non-unique) UID
  -p, --password PASSWORD       use encrypted password for the new password
  -P, --prefix PREFIX_DIR       prefix directory where are located the /etc/* files
  -r, --remove                  remove the user from only the supplemental GROUPS
                                mentioned by the -G option without removing
                                the user from other groups
  -R, --root CHROOT_DIR         directory to chroot into
  -s, --shell SHELL             new login shell for the user account
  -u, --uid UID                 new UID for the user account
  -U, --unlock                  unlock the user account
  -v, --add-subuids FIRST-LAST  add range of subordinate uids
  -V, --del-subuids FIRST-LAST  remove range of subordinate uids
  -w, --add-subgids FIRST-LAST  add range of subordinate gids
  -W, --del-subgids FIRST-LAST  remove range of subordinate gids
  -Z, --selinux-user SEUSER     new SELinux user mapping for the user account
```

- userdel:

```
leo@leo:~$ userdel --help
Usage: userdel [options] LOGIN

Options:
  -f, --force                   force some actions that would fail otherwise
                                e.g. removal of user still logged in
                                or files, even if not owned by the user
  -h, --help                    display this help message and exit
  -r, --remove                  remove home directory and mail spool
  -R, --root CHROOT_DIR         directory to chroot into
  -P, --prefix PREFIX_DIR       prefix directory where are located the /etc/* files
      --extrausers              Use the extra users database
  -Z, --selinux-user            remove any SELinux user mapping for the user
```

- su:

```
leo@leo:~$ su --help

Usage:
 su [options] [-] [<user> [<argument>...]]

Change the effective user ID and group ID to that of <user>.
A mere - implies -l.  If <user> is not given, root is assumed.

Options:
 -m, -p, --preserve-environment      do not reset environment variables
 -w, --whitelist-environment <list>  don't reset specified variables

 -g, --group <group>             specify the primary group
 -G, --supp-group <group>        specify a supplemental group

 -, -l, --login                  make the shell a login shell
 -c, --command <command>         pass a single command to the shell with -c
 --session-command <command>     pass a single command to the shell with -c
                                   and do not create a new session
 -f, --fast                      pass -f to the shell (for csh or tcsh)
 -s, --shell <shell>             run <shell> if /etc/shells allows it
 -P, --pty                       create a new pseudo-terminal

 -h, --help                      display this help
 -V, --version                   display version

For more details see su(1).
```

- groupadd:

```
leo@leo:~$ groupadd --help
Usage: groupadd [options] GROUP

Options:
  -f, --force                   exit successfully if the group already exists,
                                and cancel -g if the GID is already used
  -g, --gid GID                 use GID for the new group
  -h, --help                    display this help message and exit
  -K, --key KEY=VALUE           override /etc/login.defs defaults
  -o, --non-unique              allow to create groups with duplicate
                                (non-unique) GID
  -p, --password PASSWORD       use this encrypted password for the new group
  -r, --system                  create a system account
  -R, --root CHROOT_DIR         directory to chroot into
  -P, --prefix PREFIX_DI        directory prefix
  -U, --users USERS             list of user members of this group
      --extrausers              Use the extra users database
```

- who:

```
leo@leo:~$ who --help
Usage: who [OPTION]... [ FILE | ARG1 ARG2 ]
Print information about users who are currently logged in.

  -a, --all         same as -b -d --login -p -r -t -T -u
  -b, --boot        time of last system boot
  -d, --dead        print dead processes
  -H, --heading     print line of column headings
  -l, --login       print system login processes
      --lookup      attempt to canonicalize hostnames via DNS
  -m                only hostname and user associated with stdin
  -p, --process     print active processes spawned by init
  -q, --count       all login names and number of users logged on
  -r, --runlevel    print current runlevel
  -s, --short       print only name, line, and time (default)
  -t, --time        print last system clock change
  -T, -w, --mesg    add user's message status as +, - or ?
  -u, --users       list users logged in
      --message     same as -T
      --writable    same as -T
      --help        display this help and exit
      --version     output version information and exit

If FILE is not specified, use /var/run/utmp.  /var/log/wtmp as FILE is common.
If ARG1 ARG2 given, -m presumed: 'am i' or 'mom likes' are usual.

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Full documentation <https://www.gnu.org/software/coreutils/who>
or available locally via: info '(coreutils) who invocation'
```

- groupdel:

```
leo@leo:~$ groupdel --help
Usage: groupdel [options] GROUP

Options:
  -h, --help                    display this help message and exit
  -R, --root CHROOT_DIR         directory to chroot into
  -P, --prefix PREFIX_DIR       prefix directory where are located the /etc/* files
  -f, --force                   delete group even if it is the primary group of a user
      --extrausers              Use the extra users database
```

- passwd:

```
leo@leo:~$ passwd --help
Usage: passwd [options] [LOGIN]

Options:
  -a, --all                     report password status on all accounts
  -d, --delete                  delete the password for the named account
  -e, --expire                  force expire the password for the named account
  -h, --help                    display this help message and exit
  -k, --keep-tokens             change password only if expired
  -i, --inactive INACTIVE       set password inactive after expiration
                                to INACTIVE
  -l, --lock                    lock the password of the named account
  -n, --mindays MIN_DAYS        set minimum number of days before password
                                change to MIN_DAYS
  -q, --quiet                   quiet mode
  -r, --repository REPOSITORY   change password in REPOSITORY repository
  -R, --root CHROOT_DIR         directory to chroot into
  -S, --status                  report password status on the named account
  -u, --unlock                  unlock the password of the named account
  -w, --warndays WARN_DAYS      set expiration warning days to WARN_DAYS
  -x, --maxdays MAX_DAYS        set maximum number of days before password
                                change to MAX_DAYS
```

16. FileSystem:

(a) ¿Cómo son definidos los permisos sobre archivos en un sistema GNU/Linux?

Los archivos tienen 3 categorías de usuario a las que se le aplican permisos. El archivo pertenece a un usuario, que generalmente es quien creó el archivo. El archivo también pertenece a un solo grupo, generalmente el grupo primario del usuario que creó el archivo, pero esto se puede cambiar. Se pueden establecer diferentes permisos para el usuario propietario y el grupo propietario, así como para todos los otros usuarios en el sistema que no sean el usuario o un miembro del grupo propietario. Se aplicarán los permisos más específicos. Por lo tanto, los permisos de usuario anulan los permisos de grupo, que anulan otros permisos.

Categoría de permisos:

- Lectura/Read (r):

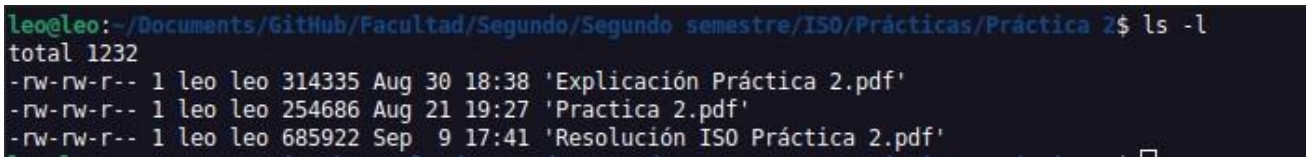
- En archivos: Pueden leerse los contenidos del archivo. ○ En directorios: Permite detallar los contenidos del directorio.

- Escritura/Write (w):

- En archivos: Pueden modificarse los contenidos del archivo. ○ En directorios: Pueden crearse o eliminar archivos en el directorio.

- Ejecución/Execute (x):

- En archivos: Se pueden ejecutar archivos como comandos.
- En directorios: Es posible acceder al contenido del directorio. El primer carácter indica el tipo del archivo:



El primer carácter indica el tipo del archivo:

- "-" para archivos normales. ○ "d" para directorios. ○ "l" para enlaces simbólicos (symlinks), entre otros.

Los siguientes 9 caracteres indican los permisos de los grupos:

- Primer grupo: Permisos del propietario. ○ Segundo grupo: Permisos del grupo. ○ Tercer grupo: Permisos para otros usuarios.

En el caso del archivo "Practica 2.pdf":

- El primer carácter "-" indica que es un archivo normal. ○ Los siguientes 3 caracteres (rw-) indican que el propietario tiene permiso de lectura y escritura.
- Los siguientes 3, (rw-) indican que el grupo tiene permiso de lectura y escritura. ○ Los últimos 3 (r--) indican que los otros usuarios tienen permiso de lectura. ○ El "1" indica el número de enlaces al archivo o directorio. ○ "leo" indica el usuario propietario.
- "leo" indica el grupo al que pertenece el archivo o directorio.
  ↳ Mismo nombre debido al "User Private Group".
- "254686" indica el peso en bytes. ○ "Aug 21 19:27" indica la fecha y hora de la última modificación. ○ "Practica 2.pdf", finalmente, indica el nombre.

(b) Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con los permisos en GNU/Linux:

- chmod
- chown ▪ chgrp

(c) Al utilizar el comando chmod generalmente se utiliza una notación octal asociada para definir permisos. ¿Qué significa esto? ¿A qué hace referencia cada valor?

* chmod (Change Mode): Se utiliza para cambiar los permisos de un archivo o directorio. En GNU/Linux, los permisos definen qué usuarios pueden leer, escribir o ejecutar un archivo o directorio.

- chmod [OPTIONS] [PERMISSIONS] FILE}

Modo simbólico: Se utilizan letras para representar los permisos:

- r para lectura (read).

- w para escritura (write).

- x para ejecución (execute).

Y utiliza la siguiente sintaxis:

- u para el usuario.

- g para el grupo.

- o para otros.

Modo numérico: Los permisos se representan con números. El sistema de permisos se basa en valores octales:

- 4 para la lectura.

- 2 para la escritura.

- 1 para la ejecución.

Estos valores se suman para asignar un permiso, por ejemplo:

Lectura+Escritura > 2+4 = 6

Lectura+Escritura+Ejecución > 2+4+1 = 7

Lectura > 2 = 4

* chown (Change Ownership): Se utiliza para cambiar el propietario (usuario) y/o el grupo asociado a un archivo o directorio.

- chown [OPTIONS] [USER][:[GROUP]] FILE

Opciones más comunes:

-R, cambia el propietario y grupo de manera recursiva en todos los subdirectorios y archivos.

--reference=archivo, cambia el propietario y grupo de un archivo utilizando los mismos que tiene otro archivo de referencia.

* chgrp (Change Group): Se utiliza para cambiar el grupo asociado a un archivo o directorio sin cambiar el propietario.

- chgrp [OPTIONS] [GROUP] FILE

(d) ¿Existe la posibilidad de que algún usuario del sistema pueda acceder a determinado archivo para el cual no posee permisos? Nombrelo, y realice las pruebas correspondientes.

El superusuario (root) tiene privilegios elevados y puede acceder a cualquier archivo o directorio en el sistema, independientemente de sus permisos.

(e) Explique los conceptos de "full path name" y "relative path name". De ejemplos claros de cada uno de ellos.

Rutas absolutas (full path name):

Una ruta absoluta es un nombre completamente calificado que comienza en el directorio (/) raíz y especifica cada subdirectorio que se atraviesa para llegar y que representa en forma exclusiva un solo archivo. Cada archivo del sistema de archivos tiene un único nombre de ruta absoluta, reconocido con una regla simple: un nombre de archivo con una barra (/) como primer carácter es el nombre de la ruta absoluta. Por ejemplo, el nombre de ruta absoluta para el archivo de registro de mensajes del sistema es /var/log/messages.

Rutas relativas (relative path name):

Al igual que una ruta absoluta, una ruta relativa identifica un archivo único y especifica solo la ruta necesaria para llegar al archivo desde el directorio de trabajo. Para reconocer nombres de ruta relativos, se sigue una regla simple: un nombre de ruta que no tenga otro carácter más que una barra (/) como primer carácter es un nombre de ruta relativo. Un usuario en el directorio /var podría referirse en forma relativa al archivo de registro del mensaje como log/messages.

./ representa al directorio de trabajo y ../ representa al directorio padre del directorio de trabajo actual. Por ejemplo, un usuario en el directorio /home/user/Documets/Foo/ puede hacer referencia al archivo /home/user/file.txt con la ruta relativa ./../../file.txt.

(f) ¿Con qué comando puede determinar en qué directorio se encuentra actualmente? ¿Existe alguna forma de ingresar a su directorio personal sin necesidad de escribir todo el path completo? ¿Podría utilizar la misma idea para acceder a otros directorios? ¿Cómo? Explique con un ejemplo.

El comando pwd (print working directory) imprime la ruta absoluta del directorio de trabajo actual. Para ingresar al directorio personal sin necesidad de escribir todo el path completo se puede utilizar "cd".

También se puede utilizar "cd ~" para acceder al directorio personal o acortar las rutas. Utilizando "~" también se puede acceder al directorio personal de otro usuario, por ejemplo:

- cd ~pepe/Downloads

↳ Accede al directorio "Downloads" del usuario "pepe"

(g) Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el uso del FileSystem:

- cd

- umount
- mkdir
- du
- rmdir
- df
- mount
- ln
- ls
- pwd
- cp
- mv

cd	Cambiar el directorio de trabajo actual.

umount	Desmontar sistemas de archivos o dispositivos.

mkdir	Crear un directorio, si no existe.	
	-m, --mode=*MODE*	Establece los permisos del directorio MODE es un octal.
	-p, --parents	Crea directorios padres según sea necesario, con sus modos de archivo no afectados por ninguna opción -m.
	-v, --verbose	Imprime cada directorio creado.

du	Muestra el uso del espacio en disco por archivos y directorios.	
	-a, --all	Muestra todos los archivos, no solo directorios.
	-d, --max-depth=*N*	Imprime los valores hasta *N* niveles.
	-h, --human-readable	Imprime los tamaños en formato legible (1K, 234M, 2G).
	-c, --total	Imprime el peso total
	`$ du -a -d -h 1 dir | sort -h`	
rmdir	Elimina un directorio, si está vacío.	
	-p, --parents	Elimina el directorio y sus ancestros. “rmdir -p a/b/c” es equivalente a “rmdir a/b/c a/b a”.
	-v, --verbose	Muestra un diagnóstico por cada directorio procesado

df	Muestra información sobre el espacio disponible y utilizado en los sistemas de archivos montados.
	-h, --human-readable Imprime los tamaños en formato legible (1K, 234M, 2G).

mount	Montar sistemas de archivos en un directorio específico

ln	Crea enlaces duros (por defecto) o simbólicos a archivos y directorios.	
	-s, --symbolic	Crea enlaces simbólicos `ln -s file link`
ls	Lista los archivos y directorios de un directorio	
	-a, --all	No ignora las entradas que empiezan con “.”
	-A, --almost-all	Como -a pero no lista ./ y ../
	-d, --directory	Lista el directorio en sí, no su contenido
	-h, --human-readable	Imprime los tamaños en formato legible.
	-l	Utilizar un formato de listado largo
	-R, --recursive	Lista los subdirectorios recursivamente

pwd	Muestra la ruta completa del directorio de trabajo actual

cp	Copia archivos y directorios de una ubicación a otra.	
	-f, --force	Si no se puede abrir un archivo de destino existente, elimínelo e inténtelo de nuevo.
	-i, --interactive	Preguntar antes de sobrescribir.
	-p	Igual a `--preserve=mode,ownership,timestamps`.
	-R, -r, --recursive	Copia directorios recursivamente.
	-v, --verbose	Explicar lo que se está haciendo.

mv	Mueve o renombra archivos y directorios.	
	-f, --force	No preguntar antes de sobrescribir
	-i, --interactive	Preguntar antes de sobrescribir.
	-v, --verbose	Explicar lo que se está haciendo.

17. Procesos:

(a) ¿Qué es un proceso? ¿A qué hacen referencia las siglas PID y PPID? ¿Todos los procesos tienen estos atributos en GNU/Linux? Justifique. Indique qué otros atributos tiene un proceso.

Un proceso es un programa que se está ejecutando.

El PID es la identificación del proceso y el PPID es la identificación de un proceso padre. En Linux todos los procesos tienen un PPID excepto por el proceso systemd o init, que son el primer proceso en ejecutarse y es de donde que derivan todos los demás procesos.

Además del PID y el PPID los procesos tienen otros atributos cómo:

- Estado: en ejecución, en espera, suspendido, zombie.
- Espacio de memoria: cada proceso tiene su propio espacio de memoria virtual.
- Información sobre la propiedad del proceso: usuario y grupo propietario del proceso.
- Prioridad de planificación: Indica la prioridad del proceso en relación con otros procesos.

(b) Indique qué comandos se podrían utilizar para ver qué procesos están en ejecución en un sistema GNU/Linux.

El comando ps se utiliza para elaborar una lista de los procesos actuales. El comando puede proporcionar información detallada de los procesos, que incluye:

- La identificación del usuario (UID) que determina los privilegios del proceso.
- La identificación del proceso (PID) única.
- La CPU y el tiempo real empleado.
- La cantidad de memoria que el proceso ha asignado en diversas ubicaciones.
- La ubicación del proceso STDOUT, conocido como terminal de control. El estado del proceso actual.

De manera predeterminada, el comando ps sin opciones selecciona todos los procesos que tienen la misma identificación de usuario efectivo (EUID) que el usuario actual y que están asociados con la misma terminal en la que se invocó ps. Una lista de visualización común es ps aux (no es lo mismo que ps -aux) que muestra todos los procesos, con columnas que serán de interés para los usuarios, e incluye procesos sin un terminal de control.

(c) ¿Qué significa que un proceso se está ejecutando en Background? ¿Y en Foreground?

En sistemas GNU/Linux y Unix, los conceptos de foreground (primer plano) y background (segundo plano) describen cómo se ejecuta un proceso con respecto al usuario y la terminal.

Foreground:

- Un proceso que se ejecuta en primer plano está activo directamente en la terminal que lo inició. El proceso ocupa la terminal y recibe directamente las entradas del usuario (teclado), además de mostrar la salida (información) directamente en la misma.

- Mientras un proceso está en primer plano, el usuario no puede interactuar con la terminal para realizar otras acciones hasta que el proceso termine.

Ejemplo: Al utilizar “cat archivo.txt” el comando está en foreground y la terminal está ocupada con la ejecución de ese comando, no se puede utilizar hasta que el proceso finalice.

Background:

- Un proceso que se ejecuta en segundo plano se está ejecutando sin bloquear la terminal, lo que permite que el usuario continúe utilizando la terminal para otros comandos mientras el proceso sigue funcionando en segundo plano.

- Un proceso en segundo plano no recibe entradas del teclado directamente, pero sigue ejecutándose y produciendo resultados.

Ejemplo: Al utilizar un comando con el símbolo “&” al final, el proceso se ejecutará en segundo plano.

(d) ¿Cómo puedo hacer para ejecutar un proceso en Background? ¿Como puedo hacer para pasar un proceso de background a foreground y viceversa?

Para ejecutar un proceso en Background se puede agregar “&” al final:

- cat archivo.txt &

Para enviar un proceso de Foreground a Background se puede utilizar Ctrl + Z para suspenderlo temporalmente y después ejecutando “bg” para enviarlo al segundo plano. Para traerlo del Background se puede utilizar “fg”.

(e) Pipe (|). ¿Cuál es su finalidad? Cite ejemplos de su utilización.

Su función es encadenar comandos, permite usar la salida de un comando como entrada de otro, y procesar datos en etapas, facilita el procesamiento de datos en varias etapas, aplicando diferentes comandos a los datos sucesivamente.

Ejemplo: ls -l | wc -l # Lista detalladamente el contenido del directorio actual y cuenta las líneas.

(f) Redirección. ¿Qué tipo de redirecciones existen? ¿Cuál es su finalidad? Cite ejemplos de utilización.

Entrada estándar, salida estándar y error estándar:

Un proceso puede necesitar leer entradas desde alguna parte y escribir salidas en la pantalla o en archivos. Un comando ejecutado desde el aviso de shell normalmente lee su entrada desde el teclado y envía su salida a su ventana de terminal.

Un proceso utiliza canales numerados denominados descriptores de archivos para obtener entradas y enviar salidas. Todos los procesos tendrán al menos tres descriptores de archivo para comenzar.

- Entrada estándar (canal 0) lee entradas desde el teclado.
- Salida estándar (canal 1) envía una salida normal al terminal.
  Error estándar (canal 2) envía mensajes de error al terminal.

Si un programa abre conexiones independientes para otros archivos, puede usar descriptores de archivo con números superiores.

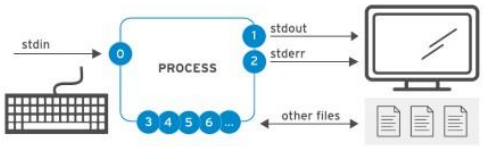


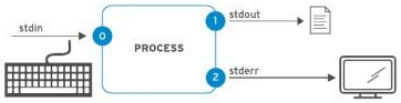
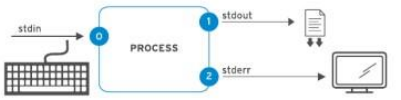
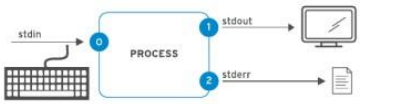
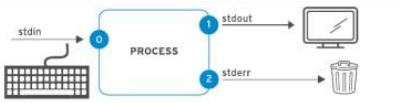
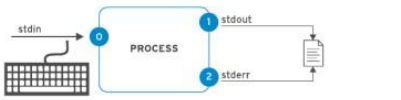
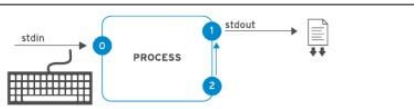
Figura 4.1: Canales de E/S de un proceso (descriptores de archivo)

#	Nombre	Descripción	Conexión predeterminada	Uso
0	**stdin**	Entrada estándar	Teclado	Solo lectura
1	**stdout**	Salida estándar	Terminal	Solo escritura
2	**stderr**	Error estándar	Terminal	Solo escritura
3+	**filename**	Otros archivos	Ninguno	Lectura y escritura

Redireccionamiento de la salida a un archivo:

El redireccionamiento de E/S reemplaza los destinos de canales predeterminados con nombres de archivos que representan dispositivos o archivos de salida. Con el uso del redireccionamiento, los mensajes de error y salida de un proceso que se envían generalmente a la ventana de terminal pueden capturarse como contenido de archivo, enviarse a un dispositivo o descartarse.

El redireccionamiento de stdout evita que la salida de un proceso aparezca en el terminal. Como se puede ver en la siguiente tabla, el redireccionamiento de únicamente stdout no evita que los mensajes de error stderr aparezcan en el terminal. Si el archivo no existe, se creará. Si el archivo existe y el redireccionamiento no es uno que se agregue al archivo, el contenido del archivo se sobrescribirá. El archivo especial /dev/null descarta discretamente la salida del canal redirigido a él y es siempre un archivo vacío.

Uso	Explicación	Ayuda visual
>file	redirigir **stdout** para sobrescribir un archivo	
>>file	redirigir **stdout** para agregar a un archivo	
2>file	redirigir **stderr** para sobrescribir un archivo	
2>/dev/null	descartar mensajes de error **stderr** mediante el redireccionamiento a **/dev/null**	
>file 2>&1 *&>file*	redirigir **stdout** y **stderr** para sobrescribir el mismo archivo	
>>file 2>&1 *&>>file*	redirigir **stdout** y **stderr** para agregar al mismo archivo	

(g) Comando kill. ¿Cuál es su funcionalidad? Cite ejemplos.

El comando kill en GNU/Linux y Unix se utiliza para enviar señales a procesos en ejecución, generalmente con el propósito de terminarlos o modificar su comportamiento. Aunque el nombre sugiere que solo sirve para “matar” procesos, puede enviar diversas señales para controlar los procesos.

Señales:

Señal	Número	Descripción	Uso común
`SIGHUP`	1	Hangup. Indica que el terminal fue cerrado o pide que se recargue la configuración.	Recargar configuraciones en procesos como `nginx` o `apache`.
`SIGINT`	2	Interrupt. Se envía cuando se presiona `Ctrl+C`.	Interrumpir un proceso en ejecución desde la terminal.
`SIGQUIT`	3	Quit. Se envía con `Ctrl+\`.	Terminar un proceso y generar un volcado de memoria (core dump).
`SIGKILL`	9	Kill. Mata un proceso inmediatamente. No puede ser ignorado.	Forzar la terminación de un proceso problemático.
`SIGTERM`	15	Terminate. Señal estándar para pedir que un proceso termine.	Terminar un proceso de manera suave y controlada.
`SIGCONT`	18	Continue. Restaura un proceso detenido con `SIGSTOP`.	Continuar un proceso que fue detenido.
`SIGSTOP`	19	Stop. Detiene un proceso sin terminarlo. No puede ser ignorado.	Pausar un proceso sin finalizarlo.
`SIGTSTP`	20	Stop signal. Se envía con `Ctrl+Z`.	Detener temporalmente un proceso en primer plano.
`SIGUSR1`	10	User-defined signal 1.	Señal definida por el usuario.
`SIGUSR2`	12	User-defined signal 2.	Señal definida por el usuario.
`SIGSEGV`	11	Segmentation fault. Indica un error de segmentación.	Generalmente usada por el sistema para indicar un error de memoria.

Ejemplos:

- kill 1234 # Envía SIGTERM al proceso con PID 1234.

- kill -9 1234 # Envía SIGKILL al proceso con PID 1234.

- kill -HUP 1234 # Envía SIGHUP al proceso con PID 1234.

- kill -u usuario # Envía SIGTERM a todos los procesos del usuario.

(h) Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el manejo de procesos en GNU/Linux. Además, compárelos entre ellos:

- ps
- kill
- pstree
- killall
- top
- nice

Comando	Funcionalidad	Parámetros principales	Comparación
`ps`	Muestra una instantánea de los procesos actuales en el sistema.	- `-e` : Muestra todos los procesos. - `-f` : Muestra una lista completa. - `-o` : Define las columnas que se muestran (como `pid`, `ppid`, `cmd`, etc.).	Proporciona una visión estática de los procesos en el momento de ejecución. Útil para listar y filtrar procesos según diferentes criterios. No muestra datos dinámicos como `top`.
`kill`	Envía señales a procesos, comúnmente para terminarlos.	- `-9` : Envía `SIGKILL`, termina un proceso inmediatamente. - `-15` : Envía `SIGTERM`, solicita terminar el proceso (por defecto). - `-l` : Lista todas las señales disponibles.	Mata procesos individualmente. Requiere el PID de un proceso, a diferencia de `killall`, que actúa sobre procesos por nombre.
`pstree`	Muestra los procesos en un formato de árbol, mostrando la jerarquía de procesos.	- `-p` : Muestra los PID junto con los nombres de los procesos. - `-u` : Muestra el propietario (usuario) de los procesos.	A diferencia de `ps`, muestra la relación padre-hijo entre los procesos, útil para visualizar cómo están relacionados jerárquicamente.
`killall`	Termina múltiples procesos que tienen el mismo nombre.	- `-9` : Mata todos los procesos con el nombre indicado. - `-I` : Ignora las diferencias entre mayúsculas y minúsculas al buscar procesos. - `-v` : Muestra más detalles sobre los procesos afectados.	Más efectivo que `kill` cuando hay varios procesos del mismo nombre que deben terminarse a la vez. No requiere conocer los PIDs específicos.
`top`	Muestra en tiempo real los procesos activos y el uso de recursos del sistema (CPU, memoria, etc.).	- `-d` : Define el tiempo de refresco (en segundos). - `-u [usuario]` : Muestra solo los procesos de un usuario específico. - `-n` : Define cuántos ciclos de actualización debe hacer antes de detenerse.	A diferencia de `ps`, que es estático, `top` proporciona una vista dinámica y actualizada en tiempo real. Es más útil para el monitoreo continuo.
`nice`	Cambia la prioridad de un proceso al iniciarlo (prioridad baja o alta).	- `-n [valor]` : Establece el valor de prioridad (`-20` es la mayor prioridad, `19` la más baja). - `-g [grupo]` : Cambia la prioridad de todos los procesos en un grupo.	Controla la prioridad de un proceso cuando se inicia, mientras que el comando `renice` puede cambiar la prioridad de procesos ya en ejecución.

18. Otros comandos de Linux (Indique funcionalidad y parámetros):

(a) ¿A qué hace referencia el concepto de empaquetar archivos en GNU/Linux?

El concepto de empaquetar archivos se refiere al proceso de agrupar varios archivos y directorios en un solo archivo comprimido o no comprimido. Esto se hace para facilitar la distribución, almacenamiento o transferencia de un conjunto de archivos. El archivo resultante, conocido como "paquete", puede ser descomprimido o desempaquetado para recuperar los archivos originales.

(b) Seleccione 4 archivos dentro de algún directorio al que tenga permiso y sume el tamaño de cada uno de estos archivos. Cree un archivo empaquetado conteniendo estos 4 archivos y compare los tamaños de los mismos. ¿Qué característica nota?

```
leo@leo:~/Desktop/Ejemplo$ ls -l
total 12
-rw-rw-r-- 1 leo leo 10240 Sep 13 17:46 archivo1.txt
-rw-rw-r-- 1 leo leo     0 Sep 13 17:43 archivo2.txt
-rw-rw-r-- 1 leo leo     0 Sep 13 17:43 archivo3.txt
-rw-rw-r-- 1 leo leo     0 Sep 13 17:43 archivo4.txt
leo@leo:~/Desktop/Ejemplo$ du --bytes
10240   .
leo@leo:~/Desktop/Ejemplo$ tar cvf archivoEmpaquetado archivo1.txt archivo2.txt archivo3.txt archivo4.txt
archivo1.txt
archivo2.txt
archivo3.txt
archivo4.txt
leo@leo:~/Desktop/Ejemplo$ ls -l
total 32
-rw-rw-r-- 1 leo leo 10240 Sep 13 17:46 archivo1.txt
-rw-rw-r-- 1 leo leo     0 Sep 13 17:43 archivo2.txt
-rw-rw-r-- 1 leo leo     0 Sep 13 17:43 archivo3.txt
-rw-rw-r-- 1 leo leo     0 Sep 13 17:43 archivo4.txt
-rw-rw-r-- 1 leo leo 20480 Sep 13 17:50 archivoEmpaquetado
leo@leo:~/Desktop/Ejemplo$ du --bytes
30720   .
leo@leo:~/Desktop/Ejemplo$
```

(c) ¿Qué acciones debe llevar a cabo para comprimir 4 archivos en uno solo? Indique la secuencia de comandos ejecutados.

(d) ¿Pueden comprimirse un conjunto de archivos utilizando un único comando?

Si se desea empaquetar y comprimir un conjunto de archivos se puede utilizar:

- tar -czvf archivo.tar archivo1 archivo2 archivo3 archivoN

- tar -cjvf archivo.tar archivo1 archivo2 archivo3 archivoN

- tar -cJvf archivo.tar archivo1 archivo2 archivo3 archivoN

Nota: j utilizará gzip (.tar.gz), z usará bzip2 (.tar.bz2) y J utiliza xz (.tar.xz).

Si ya se disponen de archivos empaquetados .tar y se desean comprimir:

- gzip archivo1.tar archivo2.tar archivo3.tar archivoN.tar - bzip2 archivo1.tar archivo2.tar archivo3.tar archivoN.tar

(e) Investigue la funcionalidad de los siguientes comandos:

- tar
- grep
- gzip
- zgrep
- wc

Comando	Funcionalidad
`tar`	Utilizado para empaquetar y descomprimir archivos. Puede combinar múltiples archivos en un solo archivo tar (`.tar`) o extraer archivos de un archivo tar. También puede comprimir archivos usando opciones adicionales como `-z` para gzip o `-j` para bzip2.
`grep`	Busca texto en archivos usando expresiones regulares. Muestra las líneas que coinciden con el patrón especificado.
`gzip`	Comprime archivos utilizando el algoritmo de compresión gzip. Cambia la extensión del archivo a `.gz` . También puede descomprimir archivos gzip usando la opción `-d` .
`zgrep`	Similar a `grep` , pero diseñado para trabajar con archivos comprimidos en formato gzip. Permite buscar texto dentro de archivos `.gz` sin necesidad de descomprimirlos primero.
`wc`	Cuenta líneas, palabras y caracteres en un archivo o en la entrada estándar. Puede mostrar diferentes estadísticas usando opciones como `-l` para contar líneas, `-w` para contar palabras y `-c` para contar caracteres.

19. **Indique qué acción realiza cada uno de los comandos indicados a continuación considerando su orden. Suponga que se ejecutan desde un usuario que no es root ni pertenece al grupo de root. (Asuma que se encuentra posicionado en el directorio de trabajo del usuario con el que se logueó). En caso de no poder ejecutarse el comando, indique la razón:**

ls −l > prueba ps >
PRUEBA chmod 710
prueba chown root
: root PRUEBA
chmod 777 PRUEBA
chmod 700 / e t c
/passwd passwd
root rm PRUEBA
man / e t c /shadow
f i n d / −name ∗.
conf usermod root
−d /home/newroot
−L cd / root
rm ∗
cd / e t c
cp ∗ /home −R

shutdown

- ls –l > prueba: Genera un listado de todos los directorios y archivos del directorio de inicio del usuario de forma detallada y lo guarda en un archivo llamado prueba.
- ps > PRUEBA: Guarda un listado con la información de los procesos que se están ejecutando actualmente en un nuevo archivo llamado PRUEBA.
- chmod 710 prueba: Cambia los permisos de acceso al archivo prueba, le da permisos de lectura, escritura y ejecución al propietario, permiso de ejecución a los participantes de su grupo y cero permisos a los demás usuarios.
- chown root : root PRUEBA: Intenta cambiar el propietario y el grupo al que pertenece el archivo PRUEBA pero no puede porque no es un usuario root.

```
mati@74706125dca4:~$ chown root:root PRUEBA
chown: changing ownership of 'PRUEBA': Operation not permitted
```

- chmod 777 PRUEBA: Cambia los permisos de acceso al archivo PRUEBA, le da permisos de lectura, escritura y ejecución al propietario, a los participantes de su grupo y a los demás usuarios.
- chmod 700 /etc/passwd: El usuario intenta cambiar los permisos de lectura, escritura y ejecución al directorio /etc/passwd pero esta operación no es posible porque solo un usuario root puede administrar los permisos de los directorios almacenados en /etc.

```
mati@74706125dca4:~$ chmod 700 /etc/passwd
chmod: changing permissions of '/etc/passwd': Operation not permitted
```

- passwd root: El usuario intenta ver o cambiar la contraseña del usuario root pero esta acción no está permitida porque solo un usuario root puede cambiar la contraseña de otros usuarios.

```
mati@74706125dca4:~$ passwd root
passwd: You may not view or modify password information for root.
```

- rm PRUEBA: Se elimina el archivo PRUEBA.
- man /etc/shadow: Esta acción no se puede realizar ya que el comando man no puede brindar un acceso a las páginas del manual del sistema ya que estas brindan información detallada sobre los comandos, utilidades, funciones del sistema y archivos del sistema operativo, no sobre directorios, además el permiso es denegado ya que intenta acceder al directorio shadow el cual contiene las contraseñas encriptadas de los usuarios.

```
mati@74706125dca4:~$ man /etc/shadow
man: can't open /etc/shadow: Permission denied
```

- find / -name *.conf: Busca y lista a partir del directorio raíz todos los archivos y directorios que contengan la extensión .conf.
- usermod root –d /home/newroot –L: El Usuario intenta cambiar el directorio de inicio del Usuario root a el directorio /home/newroot y además intenta bloquear su contraseña, esta acción no está permitida ya que el usuario no posee los permisos necesarios para ejecutar el comando.

```
mati@74706125dca4:~$ usermod root -d /home/newroot -L
bash: usermod: command not found
```

- cd /root: El usuario intenta acceder al directorio de inicio del usuario root, algo no posible ya que el usuario no posee los permisos necesarios.

```
mati@74706125dca4:~$ cd /root
bash: cd: /root: Permission denied
```

- rm *: Se eliminan todos los archivos en el directorio donde está posicionado el usuario actualmente.
- cd /etc: El usuario se sitúa en el directorio de configuraciones del sistema.
- cp * /home –R: El usuario intenta copiar todos los archivos y directorios del directorio actual /etc al directorio /home, incluyendo subdirectorios de manera recursiva, pero esto no es posible ya que solo los usuarios root tienen permisos para copiar el contenido del directorio /etc.

```
mati@74706125dca4:/etc$ cp * /home -R
cp: cannot create regular file '/home/adduser.conf': Permission denied
cp: cannot create directory '/home/alternatives': Permission denied
cp: cannot create directory '/home/apparmor.d': Permission denied
cp: cannot create directory '/home/apt': Permission denied
```

- shutdown: El usuario apaga el sistema. Se necesitan permisos sudo para hacerlo.

```
mati@74706125dca4:/etc$ shutdown
bash: shutdown: command not found
```

20. Indique qué comando sería necesario ejecutar para realizar cada una de las siguientes acciones:

(a) Terminar el proceso con PID 23.

Se podría utilizar los comandos “kill 23” ó “kill -9 23”.

(b) Terminar el proceso llamado init o systemd. ¿Qué resultados obtuvo?

Se podría intentar usar el comando “kill -9 1” pero no se obtendría ningún resultado ya que el proceso init es de suma importancia para el arranque del sistema y este no puede ser terminado.

(c) Buscar todos los archivos de usuarios en los que su nombre contiene la cadena “.conf”

Se podría utilizar el comando “find /home -name *.conf*”.

(d) Guardar una lista de procesos en ejecución el archivo **/home/<su nombre de usuario>/procesos**

Se podría utilizar el comando “ps > /home/mati/procesos”.

(e) Cambiar los permisos del archivo **/home/<su nombre de usuario>/xxxx** a:

- Usuario: Lectura, escritura, ejecución
- Grupo: Lectura, ejecución
- Otros: ejecución

Se podría utilizar el comando “chmod 751 /home/suUsuario/archivoX”.

(f) Cambiar los permisos del archivo **/home/<su nombre de usuario>/yyyy** a:

- Usuario: Lectura, escritura.
- Grupo: Lectura, ejecución
- Otros: Ninguno

Se podría utilizar el comando “chmod 650 /home/suUsuario/archivoY”.

(g) Borrar todos los archivos del directorio **/tmp** m /tmp/* # Solo borra archivos. rm -r /tmp/* # Borra archivos, links y directorios.

(h) Cambiar el propietario del archivo **/opt/isodata** al usuario **iso2010**

Se podría utilizar el comando “sudo chown iso2010 /opt/isodata”.

(i) Guardar en el archivo **/home/<su nombre de usuario>/donde** el directorio donde me encuentro en este momento, en caso de que el archivo exista no se debe eliminar su contenido anterior.

Se podría utilizar el comando “pwd >> /home/usuario/donde”.

21. Indique qué comando sería necesario ejecutar para realizar cada una de las siguientes acciones:

(a) Ingrese al sistema como usuario “root” su - root

(b) Cree un usuario. Elija como nombre, por convención, la primera letra de su nombre seguida de su apellido. Asígnele una contraseña de acceso.

useradd -m -s /bin/bash MGuaymas # Sin contraseña porque debe ingresarse encriptada.

passwd Blacky # Se le asigna una contraseña.

(c) ¿Qué archivos fueron modificados luego de crear el usuario y qué directorios se crearon?

Se modificaron los archivos /etc/passwd, /etc/shadow, /etc/group, /etc/gshadow y se creó el directorio de inicio del nuevo usuario en /home/MGuaymas. **(d)** Crear un directorio en /tmp llamado cursada2017 mkdir /tmp/cursada2017

(e) Copiar todos los archivos de /var/log al directorio antes creado. sudo cp /var/log/* > /tmp/cursada2017.

(f) Para el directorio antes creado (y los archivos y subdirectorios contenidos en él) cambiar el propietario y grupo al usuario creado y grupo users. chown -R MGuaymas:users cursada2017/

(g) Agregue permiso total al dueño, de escritura al grupo y escritura y ejecución a todos los demás usuarios para todos los archivos dentro de un directorio en forma recursiva.

chmod -R 723 cursada2017/

(h) Acceda a otra terminal virtual para loguearse con el usuario antes creado.

su - MGuaymas

(i) Una vez logueado con el usuario antes creado, averigüe cuál es el nombre de su terminal. cat /etc/passwd | grep MGuaymas | cut -f7 -d:

cat /etc/passwd - Concateno el contenido del archivo.

grep MGuaymas - Filtro por el usuario que deseo.

cut -f7 -d: - Tomo la columna 7 y delimito el

contenido por ":" **(j)** Verifique la cantidad de

procesos activos que hay en el sistema.

ps -e | wc -l

(k) Verifiqué la cantidad de usuarios conectados al sistema.

who

(l) Vuelva a la terminal del usuario root, y envíele un mensaje al usuario anteriormente creado, avisándole que el sistema va a ser apagado.

su.

wall "El sistema está por apagarse". # Envía un mensaje a todos los usuarios que están actualmente conectados al sistema, se necesita permisos sudo para usarlo.

(m) Apague el sistema.

shutdown -P now

22. Indique qué comando sería necesario ejecutar para realizar cada una de las siguientes acciones:

(a) Cree un directorio cuyo nombre sea su número de legajo e ingrese a él.

mkdir 23061_0

(b) Cree un archivo utilizando el editor de textos vi, e introduzca su información personal: Nombre, Apellido, Número de alumno y dirección de correo electrónico. El archivo debe llamarse "LEAME".

vi LEAME

Dentro del editor cambiar a modo de inserción usando la tecla "i", escribir toda la información en el archivo y luego volver al modo comando usando la tecla "Esc".

Una vez en el modo comando escribir el comando ":wq".

(c) Cambie los permisos del archivo LEAME, de manera que se puedan ver reflejados los siguientes permisos:

- Dueño: ningún permiso
- Grupo: permiso de ejecución ▪ Otros: todos los

permisos chmod 017 LEAME

(d) Vaya al directorio /etc y verifique su contenido. Cree un archivo dentro de su directorio personal cuyo nombre sea leame donde el contenido del mismo sea el listado de todos

los archivos y directorios contenidos en /etc. ¿Cuál es la razón por la cual puede crear este archivo si ya existe un archivo llamado “LEAME” en este directorio?

“cd /etc”.

“ls > /home/leame”.

Esta acción se puede realizar debido a que GNU/Linux es case sensitive.

(e) ¿Qué comando utilizaría y de qué manera si tuviera que localizar un archivo dentro del filesystem? ¿Y si tuviera que localizar varios archivos con características similares? Explique el concepto teórico y ejemplifique.

Para buscar un solo archivo con find: “find / -name nombre_del_archivo”.

Para buscar archivos con características similares: “find / - name “*cadena_a_contener”.

(f) Utilizando los conceptos aprendidos en el punto e), busque todos los archivos cuya extensión sea .so y almacene el resultado de esta búsqueda en un archivo dentro del directorio creado en a). El archivo deberá llamarse .ejercicio_f".

find / -type f -name *.so > /home/suUsuario/legajo/.ejercicio_f # Solo archivos.

find / -name *.so > /home/suUsuario/legajo/.ejercicio_f # Archivos, directorios y links.

23. Indique qué acción realiza cada uno de los comandos indicados a continuación considerando su orden. Suponga que se ejecutan desde un usuario que no es root ni pertenece al grupo de root. (Asuma que se encuentra posicionado en el directorio de trabajo del usuario con el que se logueó). En caso de no poder ejecutarse el comando indique la razón:

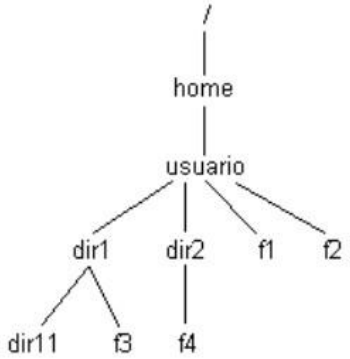
```
mkdir iso
cd ./iso; ps > f0
ls > f1
cd /
echo $HOME
ls -l &> $HOME/iso/ls
cd $HOME; mkdir f2
ls -ld f2
chmod 341 f2
touch dir
cd f2
cd ~/iso
pwd >f3
ps | grep 'ps' | wc -l >> ../f2/f3
chmod 700 ../f2; cd ..
find . -name etc/passwd
find / -name etc/passwd
mkdir ejercicio5
-------------------------------------------------
-------------------------------------------------
```

(a) Inicie 2 sesiones utilizando su nombre de usuario y contraseña. En una sesión vaya siguiendo paso a paso las órdenes que se encuentran escritas en el cuadro superior. En la otra sesión, cree utilizando algún editor de textos un archivo que se llame .ejercicio10_explicacion" dentro del directorio creado en el ejercicio 9.a) y, para cada una de las órdenes que ejecute en la otra sesión, realice una breve explicación de los resultados obtenidos.

(b) Complete en el cuadro superior los comandos 19 y 20, de manera tal que realicen la siguiente acción:

- 19: Copiar el directorio iso y todo su contenido al directorio creado en el inciso 9.a).
- 20: Copiar el resto de los archivos y directorios que se crearon en este ejercicio al directorio creado en el ejercicio 9.a).

(c) Ejecute las órdenes 19 y 20 y coméntelas en el archivo creado en el inciso a).



- "mkdir iso": Crea el directorio "iso" en su directorio inicial.
- "cd ./iso ; ps > f0": Se sitúa en el directorio "iso" y crea el archivo "f0" con un listado de todos los procesos en ejecución en el sistema actualmente.
- "ls > f1": Crea un archivo llamado "f1" en el directorio "iso" que contiene la lista de archivos y/o subdirectorios del directorio "iso".
- "cd /": Se sitúa en el directorio raíz.
- "echo $HOME": Muestra en la salida estándar la dirección del directorio inicial del usuario.
- "ls –l &> $HOME/iso/ls": Se crea un nuevo archivo llamado "ls" en el directorio "iso" ubicado en el directorio inicial del usuario que va a tener un listado detallado de los archivos y/o directorios del directorio raíz y además también la salida de error del comando realizado "ls –l", esto se logra sumando a la redirección el símbolo "&".
- "cd $HOME ; mkdir f2": Se sitúa en su directorio inicial y crea el directorio "f2".
- "ls –ld f2": Muestra información detallada del directorio "f2". Se especifica que se trate al mismo como un directorio con el parámetro "d".
- "chmod 341 f2": Cambia los permisos de lectura, escritura y ejecución del directorio "f2". Escritura y Ejecución para el dueño, Lectura para el grupo y Ejecución para los demás usuarios.
- "touch dir": Crea un archivo llamado "dir" en el directorio inicial del usuario.
- "cd f2": Se sitúa en el directorio "f2".
- "cd ~/iso": Se sitúa en el directorio "iso".
- "pwd > f3": Crea un archivo nuevo en el directorio "iso" que contiene la información de la ruta actual del usuario en ese momento "~/iso".
- "ps | grep " ps " | wc –l >> ../f2/f3": Se listan todos los procesos en ejecución y el "grep" recibe esa lista como entrada estándar, allí filtra los resultados por aquellos procesos que posean la cadena "ps", ese nuevo listado filtrado es pasado como entrada estándar al comando "wc" el cuál cuenta la cantidad de líneas de ese

listado, es decir, la cantidad de procesos que se encontraron con la cadena “ps” para luego redirigir esa salida al archivo “f3” añadiendo la cantidad al final del mismo.

- “chmod 700 ../f2 ; cd ..”: Se modifican los permisos de lectura, escritura y ejecución del directorio “f2”. Lectura, Escritura y Ejecución para el dueño y cero permisos para el grupo y otros usuarios. Luego al ejecutar “cd ..” escala un nivel hacia arriba en la estructura de directorios, ahora situándose nuevamente en su directorio inicial.
- “find . –name etc/passwd”: Se muestra un mensaje de warning por el mal uso del comando “find”.
- “find / -name etc/passwd”: Muestra en pantalla el listado de los archivos y/o directorios de “/etc/passwd” ordenados por nombre.
- “mkdir ejercicio5”: Crea el directorio “ejercicio5”.

24. Indique qué comando/s es necesario para realizar cada una de las acciones de la siguiente secuencia de pasos (considerando su orden de aparición):

(a) Cree un directorio llamado logs en el directorio /tmp.

sudo mkdir /tmp/logs

(b) Copie todo el contenido del directorio /var/log en el directorio creado en el punto anterior.

sudo cp -a /var/log/* /tmp/logs # -r por si hay directorios.

(c) Empaquete el directorio creado en 1, el archivo resultante se debe llamar "misLogs.tar".

tar –cvf misLogs.tar logs

(d) Empaquete y comprima el directorio creado en 1, el archivo resultante se debe llamar "misLogs.tar.gz". tar –cvfz misLogs.tar.gz logs

(e) Copie los archivos creados en 3 y 4 al directorio de trabajo de su usuario. cp misLogs.tar misLogs.tar.gz /home/suUsuario

(f) Elimine el directorio creado en 1, logs.

rm -r logs

(g) Desempaquete los archivos creados en 3 y 4 en 2 directorios diferentes.

mkdir /home/suUsuario/1 /home/suUsuario/2 tar -

xvf misLogs.tar -C /home/suUsuario/1 # -C para

cambiar la ruta.

tar -xzvf misLogs.tar.gz -C /home/suUsuario/2