TRABAJO PRACTICO 1   
Introducción a los Sistemas Operativos

**1) Caracteristicas de GNU/Linux**

**a) Mencione y explique las características mas relevantes de GNU/Linux**

GNU/Linux es un software libre que posee las siguientes características:

* Puede ser usado, copiado, estudiado, modificado y redistribuido libremente.
* Generalmente es de costo nulo. Es un gran error asociar software libre con el software gratuito. Puede haber softwares gratuitos con restricciones para distribuir.
* Es común que se distribuya junto con el código fuente.
* Rápida corrección ante fallas.
* Los usuarios tienen libertad para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el sofware.

**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 MacOs son de software propietario por lo cual:

* Tienen un costo asociado
* No se puede distribuir de manera libre
* No permiten su modificación
* Normalmente no se distribuye con su código fuente
* La corrección de fallas está a cargo del propietario (se deben esperar las actualizaciones)
* Menos necesidad de técnicos asociados.

**c) Que es GNU?**

**GNU** significa *“GNU’s Not Unix”* (“GNU no es Unix”), un juego de palabras que indica que es similar a Unix pero completamente libre. El sistema GNU incluye todo el software GNU, además de muchos otros paquetes.

Fue iniciado en **1983 por Richard Stallman**, con el objetivo de crear un sistema operativo completamente **libre**, es decir, que cualquiera pueda **usar, estudiar, modificar y redistribuir**.

GNU proporciona **herramientas esenciales de un sistema operativo**, como:

* Compiladores (*GCC*),
* Editores (*Emacs*),
* Shells (*Bash*),
* Librerías básicas (*glibc*),
* Utilidades para manejo de archivos y procesos.

Nota: GNU cubre casi todo el sistema, **excepto el núcleo (kernel)**.

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

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

Fue iniciado por Richard Stallman, con el fin de que crear Unix libre (El sistema gnu). En 1985, Stallman creo la Free Software Foundation (FSF) para proveer soportes lógicos, legales y financieros al proyecto GNU. Esta fundación contrato programadores, aunque una porción sustancial del desarrollo fue y sigue siendo producida por voluntarios.

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

Linus Torvalds ya venia trabajando desde 1991 en un Kernel denominado Linux, el cual se distribuye 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 cono GNU/Linux o como distribución Linux y existen diversas variantes.

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

GNU/Linux hace uso de la multitarea, multiusuario y multiprocesador. Multitarea significa que le permite al usuario realizar varias tareas al mismo tiempo. Este puede estar editando el código fuente de un programa durante su depuración compila otro programa, a la vez que esta recibiendo un correo electrónico en un proceso background.

**f) Que es POSIX?**

POSIX cosiste 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 un software que cumple con los estándares POSIX existe una gran probabilidad de que se podrá usar en SO del tipo Linux. Si se ignoran tales reglas, es muy probable 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 SO basado en GNU/Linux. Son “sabores” de GNU/Linux que, en general, se diferencian entre si por las herramientas para la configuración y sistemas de administración de paquetes de software para instalar. La elección de una distribución depende de las necesidades de usuario y de gustos personales.

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

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



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 repositorios de 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 mas al desarrollo del sofware y servidores
* **Android:** es un sistema operativo desarrollado por Google, baso en el Kernel de Linux y otros softwares de código abierto. Fue desarrollado para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tabletas, relojes inteligentes, autos y televisores.

**b) En que se diferencia una distribución de otra?**

Las distribuciones de Linux tienen en común el Kernel, pero el resto de los componentes (las herramientas, la Shell, el Display Server, la GUI) varían entre si, se personalizan o se crean desde cero, por eso las distribuciones son tan diferentes entre si. 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 un sistema 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 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) Que es Debian? Acceda al sitio en indique cuales son los objetivos del proyecto y una breve cronología de este.**

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 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.

**3. Estructura de GNU/Linux**

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

* Kernel (núcleo)
* El Shell (interprete de comandos)
* El FileSystem (Sistema de archivos)

**b) Mencione y explique la estructura básica del SO 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.

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.

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.

**4. Kernel**

**a) Que 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 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 totalmente 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) Cuales son sus funciones principales?**

Sus funciones mas 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 mas de un Kernel de GNU/Linux instalado en la misma máquina?**

Se puede tener mas de un Kernel instalado (versión anterior o algún Kernel virtual), pero en realidad estaría funcionando solo uno, ya que es la parte de un sistema operativo que adminsitra y controla 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 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. Interprete de comandos (Shell)**

**a) Que es?**

Es el programa que recibe lo que se escribe en la terminal y lo convierte en instrucciones para el SO. 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) Cuales son sus funciones?**

Actúa como interfaz para comunicar al usuario con el SO mediante una ventana que espera comandos textuales ingresados por el usuario del 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 mas instrucciones. Se interactúa con la información de la manera mas simple posible, sin graficas, solo texto.

**c) Mencione al menos tres interpretes 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 las 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) Donde 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 se usa el comando 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 de el 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), se suelen encontrar en el directorio /usr/binLos externos se encuentran en las variables $PATH, no son nativos del shell de Linux. Ser capaz de editar tu path es una habilidad importante para todo principiante de Linux.

**e) Por que se 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) Que 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 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 y los pendrives USB, desde aquí puedes acceder a sus archivos.

/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

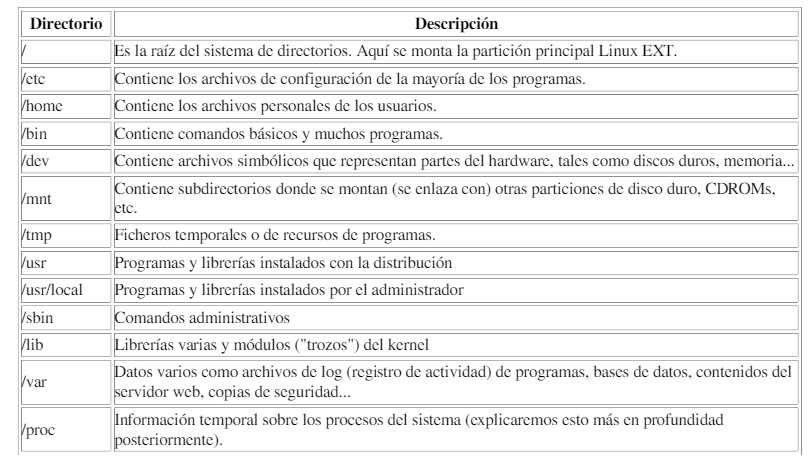
/root: La carpeta home para el usuario root.

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

/tmp: Contiene los archivos temporale

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

/var: Contiene registros del sistema y otros datos variable



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 desventajas:

* Es una buena práctica separar los datos del usuario de las aplicaciones y/o SO 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) Como se identifican las particiones en GNU/Linux? Considere discos, IDE, SCSI y SATA.**

* **Disqueteras:**
* Primera disquetera: /dev/fd0 (en Windows seria 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/hdb5
    - Sucesivamente: /dev/hdb#
* **Discos SCSI**
  + Primer disco SCSI: /dev/sda
  + Segundo disco SCSI: /dev/sbd
    - Sucesivamente…

**Primer CD\_ROM SCSI:** /dev/scd0, también conocido como /dev/sr0

**c) Cuantas particiones son necesarias como mínimo para instalar GNU/Linux? ¿Nómbrelas indicando tipo de partición, identificación, tipo de FS 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 un SO**

Algunos sistemas de archivos (p.e versiones antiguas de sistemas FAT de Microsoft) tienen tamaños máximos mas 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, videos, documentos), para que al momento de reinstalar algún SO 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 dos tipos de software para particionar:

* Destructivos: Permiten crear y eliminar particiones (fdisk)
* No destructivos: 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» tambiénpuede ajustar el tamaño de las particiones, crear sistemas de ficheros (como se llamaen 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, que es un nexo entre el sistema operativo y el firmware. Desde este punto de vista puede verse 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.**

UEFI utiliza 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 formatode los directorios. 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 y editores:**

**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:

* Puede tener entre 1 y 255 caracteres.
* Se puede utilizar cualquier carácter especial excepto la barra inclinada “/” y no es recomendable emplear caracteres con significado especial en Linux (=, ^,~,’,”,´\*……). Para emplear ficheros con esos caracteres o espacios hay que introducir el nombre del fichero con comillas.
* Se pueden usar números exclusivamente si así lo desea.
* Las letras mayúsculas y minisculas se consideran diferentes, y lo tanto no es lo mismo carta.txt que Carta.txt o carcta.Txt.

Como en Windows se puede emplear cierto criterio de “tipo” para marcar las distintas clases de ficheros empleado una serie de caracteres al final del nombre que indiquen el tipo de fichero del que se trata. Asi, los ficheros de texto, HTML, PONG o JPEG tienen extensiones .txt, .html, .png, .jpeg respectivamente. Pese a esto Linux solo distingue tres tipos de archivos:

* Archivos o ficheros ordinarios, son los mencionados anteriormente.
* Directorios (o cartepeas), 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 tiene 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.**

**Editores 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 al editor cada vez que se inicia. Las teclas se 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 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.

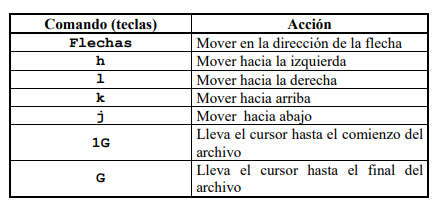
INICIO DE VI:

vi: Abre la ventana de edición sin abrir ningún archivo.

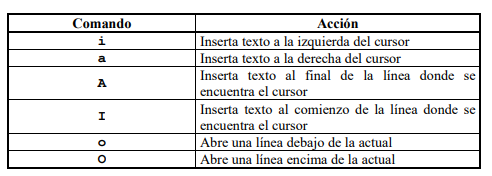
vi archivo1: Edita el archivo archivo1 si ya existe, de lo contrario, lo crea. Evidentemente se debe indicar el camino (path) que conduce al archivo (si existe) o el camino que conduce al directorio donde se desea crear el archivo (si no existe)

**MODO COMANDO:**

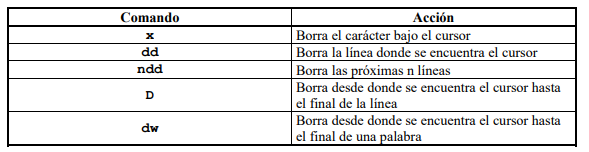
El editor vi, como todo UNIX, diferencia entre mayúsculas de minúsculas. A continuación, se comentan algunos comandos útiles en el manejo del editor.

**Movimiento del cursor**

**Cambio de modo comando a texto:**



**Borrar Texto:**



Es importante destacar que todo lo que se borra queda almacenado en un buffer (área temporal de memoria), de modo que si se borro 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 eso 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 usados 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 ultimo cambio realizado con el comando u

**MODO TEXTO:**

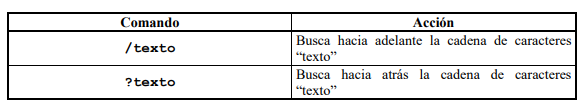
En este modo se ingresa el texto deseado. Para pasar de modo texto a modo comando simplemente se debe apretar la tecla ESC

**MODO LINEA**

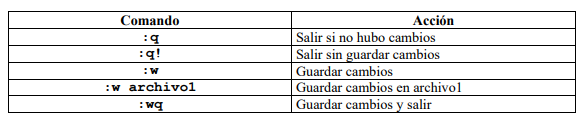
Para ingresar al modo línea desde el modo comando, se debe utilizar alguna de las siguientes teclas: /, ?, :

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

**Buscar texto:**



Salir de VI, salvar, no salvar cambios, etc;



**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.

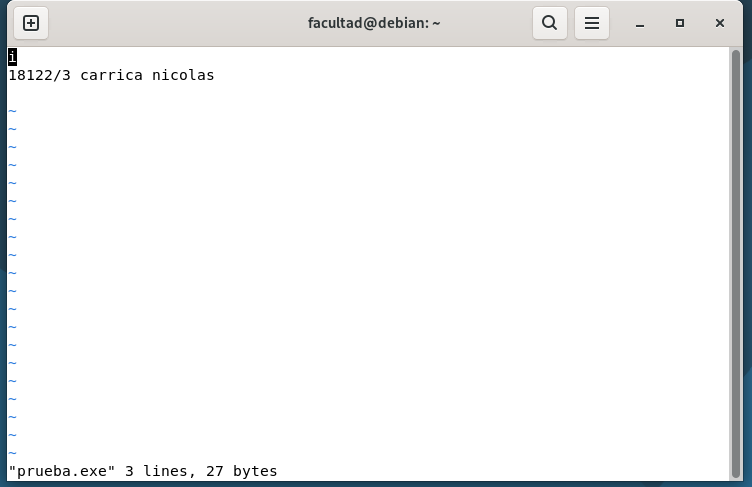
Incluye un editor llamado mcedit. Mcedit es un ejecutable independiente, el cual también puede ser usado de forma independiente a Midnight Commander. Esta aplicación permite visualizar el contenido de ficheros y disfrutar de características como la de resaltar la sintaxis para ficheros de código fuente 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 estándaRr.

**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 sirve para identificar el tipo de archivo. No se fija en la extensión (como .txt o .exe), sino que lee el contenido interno del archivo y te dice de qué tipo es.

**e) Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el uso de archivos:**

**i. cd:** Sirve para cambiar de directorio, Ejemplo:

* cd Documentos → entro a la carpeta Documentos.
* cd .. → vuelvo un nivel atrás.

**ii. mkdir:** Crea una nueva carpeta.

* Ejemplo: mkdir tareas → crea la carpeta llamada tareas.

**iii. rmdir:** Borra una carpeta vacía.

* **Ejemplo:** rmdir tareas. Si tiene archivos adentro, no la borra (para eso se usa rm -r).

**(iv) ln:** El comando ln crea enlaces entre archivos. Existen dos tipos:

* **Enlace duro (hard link):** apunta directamente al contenido.
* **Enlace simbólico (soft link o symlink):** es como un acceso directo.

**Ejemplo:**

ln archivo.txt copia.txt → crea un enlace duro llamado copia.txt.

ln -s archivo.txt acceso.txt → crea un enlace simbólico llamado acceso.txt.

**v. tail:** Muestra las últimas líneas de un archivo de texto.Ejemplo: tail archivo.log.

* **Con tail -n 20 archivo.log muestra las últimas 20 líneas.**
* **Con tail -f archivo.log se queda “escuchando” lo que se agregue en tiempo real.**

**vi. locate:** Sirve para buscar un archivo en todo el sistema.

* **Ejemplo: locate prueba.exe.**
* **Usa una base de datos que se actualiza con updatedb.**

**vii. ls:** Lista los archivos y carpetas del directorio actual.

* **Ejemplo: ls.**
* **Con ls -l muestra detalles (permisos, fecha, tamaño).**
* **Con ls -a también muestra los ocultos.**

**viii. pwd:** Muestra en qué carpeta estoy ubicado.

* **Ejemplo: /home/usuario/Documentos.**

**ix. cp:** Sirve para copiar archivos o carpetas.

* **Ejemplo: cp archivo.txt copia.txt.**
* **Para carpetas se usa cp -r carpeta1 carpeta2.**

**x. mv:** Sirve para mover o renombrar archivos.

* **Ejemplo: mv archivo.txt /home/usuario/.**
* **También se usa para renombrar: mv viejo.txt nuevo.txt.**

**xi. find:** Busca archivos en carpetas, recorriendo todo.

* **Ejemplo: find . -name "archivo.txt" → busca en el directorio actual (.).**
* **Ejemplo: find /home -type d -name "tareas" → busca una carpeta llamada tareas.**

**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 ISOCSO.**

* mkdir ISOCSO

**b) Acceda a la carpeta:**

* cd ISOCSO

**c) Cree dos archivos con los nombres isocso.txt e isocso.csv**

* touch isocso.txt isocso.csv

**d) Liste el contenido del directorio actual**

* ls

**e. Visualizar la ruta donde estoy situado**

* pwd

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

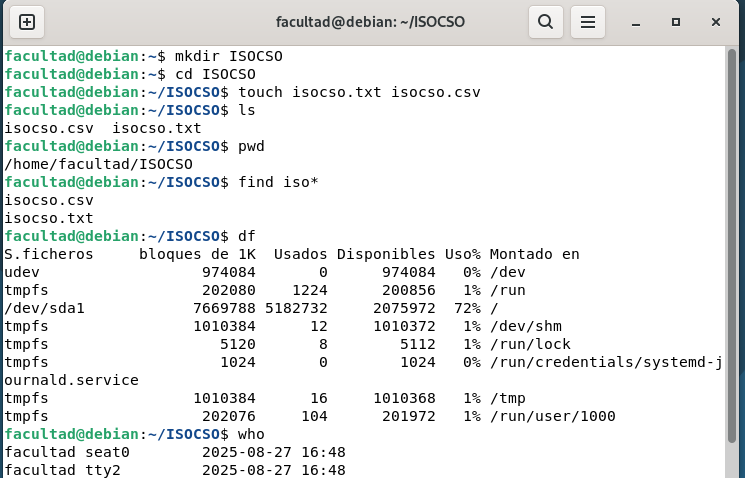
* find iso\*

**g. Informar la cantidad de espacio libre en disco**

* df

**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



**(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]

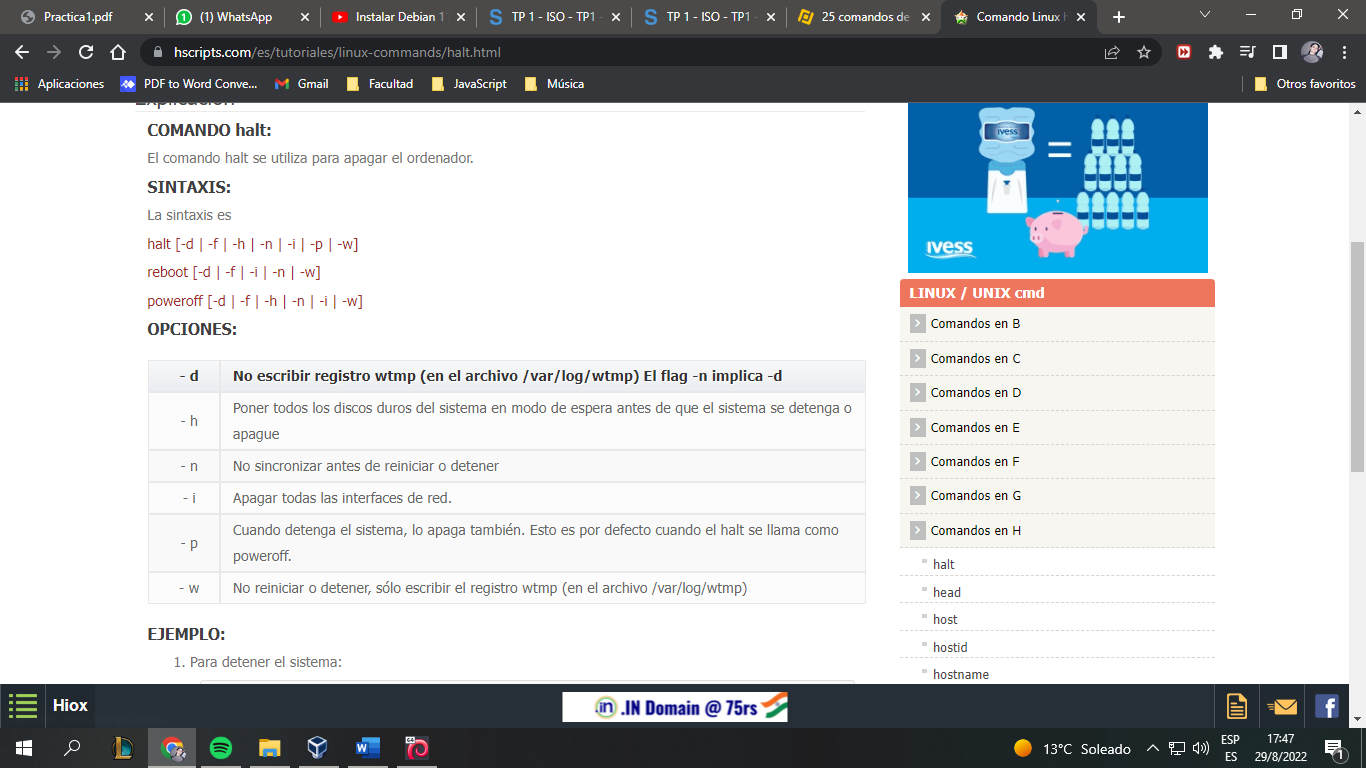
**(b) reboot**

Reinicia SO

-reboot, -r: reinicia el sistema

**(c) halt**

El comando halt se utiliza para apagar el ordenador



**(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-administrar-particiones-de-disco-en-gnulinux/

**12. Procesos.**

**a. ¿Qué es un proceso? ¿A que 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 en ejecución, es decir, una instancia activa de un programa que está cargada en memoria y gestionada por el sistema operativo. Incluye el código del programa, los datos, el contador de programa, los registros de CPU, la pila y otros recursos que necesita para ejecutarse.

**PID y PPID**

* **PID (Process ID):** es el identificador único que el sistema operativo asigna a cada proceso cuando se crea. Permite distinguirlo de otros procesos en ejecución.
* **PPID (Parent Process ID):** es el identificador del proceso padre, es decir, del proceso que creó al proceso actual.

**¿Todos los procesos tienen estos atributos en GNU/Linux?**  
Sí. En GNU/Linux **todo proceso tiene un PID y un PPID**:

* El **PID** es imprescindible porque el kernel necesita identificar y controlar a cada proceso de forma unívoca.
* El **PPID** también siempre está presente: incluso el primer proceso del sistema, systemd (antes init), que tiene **PID 1**, actúa como padre de todos los procesos. Si un proceso “pierde” a su padre porque este finaliza, automáticamente el kernel le reasigna como padre al proceso 1.

Por lo tanto, **todos los procesos en GNU/Linux poseen PID y PPID**.

**Otros atributos de un proceso en GNU/Linux**  
Además del PID y PPID, un proceso posee muchos más atributos, entre ellos:

* **UID y EUID:** identificadores del usuario propietario del proceso (real y efectivo).
* **GID y EGID:** identificadores del grupo propietario (real y efectivo).
* **Estado del proceso:** puede estar en ejecución, esperando, detenido, zombie, etc.
* **Prioridad y valor nice:** usados por el planificador para decidir el orden de ejecución.
* **Contador de programa (PC):** indica la próxima instrucción a ejecutar.
* **Uso de CPU y memoria:** cuánto consume de recursos.
* **Tabla de archivos abiertos (FDs):** lista de descriptores de archivos, sockets, pipes asociados al proceso.
* **Áreas de memoria asignadas:** código, datos, pila, heap.
* **Tiempo de inicio y tiempo de CPU acumulado.**

**b) Investigue el funcionamiento, parámetros y ubicación (directorio) de los siguientes comandos relacionados a procesos. En caso de que algún comando no venga por defecto en la distribución que utiliza deberá proceder a instalarlo**

**i. top**

* **Función:** Muestra en tiempo real los procesos en ejecución, consumo de CPU, memoria, carga del sistema, etc.
* **Parámetros más comunes:**
  + -u <usuario>: filtra procesos por usuario.
  + -p <PID>: muestra solo un proceso específico.
  + -n <N>: cantidad de actualizaciones antes de salir.
* **Ubicación:** /usr/bin/top

**ii. htop *(no siempre instalado por defecto)***

* **Función:** Similar a top pero con interfaz interactiva y más amigable. Permite matar procesos con teclas, ver jerarquía, ordenar por consumo.
* **Parámetros más comunes:**
  + -u <usuario>: filtra procesos por usuario.
  + -p <PID>: abre mostrando un proceso específico.
* **Ubicación:** /usr/bin/htop

**iii. ps**

* **Función:** Muestra información estática de procesos (no en tiempo real).
* **Parámetros más comunes:**
  + aux: lista todos los procesos en ejecución con detalles.
  + -e o -A: todos los procesos.
  + -f: formato extendido, muestra jerarquía.
* **Ubicación:** /bin/ps

**iv. pstree *(a veces no instalado por defecto)***

* **Función:** Muestra los procesos en forma de árbol, indicando relaciones padre-hijo.
* **Parámetros más comunes:**
  + -p: incluye el PID.
  + -u: incluye el usuario.
* **Ubicación:** /usr/bin/pstree

**v. kill**

* **Función:** Envía una señal a un proceso (por defecto SIGTERM).
* **Parámetros más comunes:**
  + kill <PID>: termina el proceso con SIGTERM.
  + kill -9 <PID>: fuerza la finalización (SIGKILL).
  + kill -l: lista todas las señales.
* **Ubicación:** /bin/kill

**vi. pgrep**

* **Función:** Busca procesos por nombre y devuelve su PID.
* **Parámetros más comunes:**
  + pgrep <nombre>: devuelve PIDs de procesos con ese nombre.
  + -u <usuario>: filtra por usuario.
  + -l: muestra también el nombre del proceso.
* **Ubicación:** /usr/bin/pgrep

**vii. pkill**

* **Función:** Mata procesos por nombre o patrón (más amigable que usar kill con PID).
* **Parámetros más comunes:**
  + pkill <nombre>: termina procesos por nombre.
  + pkill -9 <nombre>: fuerza la terminación.
  + -u <usuario>: limita al usuario.
* **Ubicación:** /usr/bin/pkill

**viii. killall**

* **Función:** Mata todos los procesos que coinciden con un nombre.
* **Parámetros más comunes:**
  + killall <nombre>: mata procesos con ese nombre.
  + killall -9 <nombre>: mata forzadamente.
  + -u <usuario>: afecta solo procesos de un usuario.
* **Ubicación:** /usr/bin/killall

**ix. renice**

* **Función:** Cambia la prioridad (valor *nice*) de un proceso ya en ejecución.
* **Parámetros más comunes:**
  + renice <prioridad> -p <PID>: cambia prioridad de un proceso.
  + renice <prioridad> -u <usuario>: cambia procesos de un usuario.
* **Ubicación:** /usr/bin/renice

**x. xkill *(no siempre instalado por defecto, requiere entorno gráfico)***

* **Función:** Permite finalizar un proceso gráfico haciendo clic sobre su ventana.
* **Parámetros más comunes:**
  + Sin parámetros: cambia el puntero del mouse para seleccionar ventana.
* **Ubicación:** /usr/bin/xkill

**xi. atop *(no siempre instalado por defecto)***

* **Función:** Similar a top, pero mucho más completo: muestra estadísticas detalladas de CPU, memoria, disco, red y puede generar reportes históricos.
* **Parámetros más comunes:**
  + -r <archivo>: lee un log guardado.
  + -w <archivo>: guarda el monitoreo en un archivo binario.
* **Ubicación:** /usr/bin/atop

**xii. nice**

* **Función:** Ejecuta un programa con una prioridad determinada (valor *nice*).
* **Parámetros más comunes:**
  + nice -n <valor> comando: ejecuta un comando con la prioridad indicada.
  + Valores van de -20 (alta prioridad) a 19 (baja prioridad).
* **Ubicación:** /usr/bin/nice

**13 - Proceso de Arranque SystemV (**[**https://github.com/systeminit/si**](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**

El proceso de arranque en un sistema GNU/Linux con **init System V** se puede dividir en varias etapas:

**1) Encendido y POST (Power-On Self Test)**

* La BIOS/UEFI realiza un chequeo del hardware (memoria, CPU, discos, periféricos).
* Busca un dispositivo de arranque (según el orden configurado en BIOS/UEFI).

**2) Carga del Bootloader (GRUB, LILO, etc.)**

* El cargador de arranque se localiza en el MBR (Master Boot Record) o partición EFI.
* Permite seleccionar qué sistema operativo arrancar.
* Carga el kernel de Linux en memoria junto con la **imagen initrd/initramfs**.

3) **Inicialización del Kernel de Linux**

* El kernel se descomprime y comienza a ejecutarse.
* Detecta y configura el hardware básico (controladores de disco, red, USB, etc.).
* Monta el **sistema de archivos raíz inicial** (initrd/initramfs).
* Ejecuta el **primer proceso en espacio de usuario**: init (PID 1).

**4) Ejecución del proceso init (System V Init)**

* init es el proceso padre de todos los procesos en espacio de usuario.
* Lee su archivo de configuración /etc/inittab.
* Determina el **runlevel** de arranque (modo de operación del sistema).

**5) Ejecución de scripts de arranque**

* Según el runlevel, init ejecuta scripts ubicados en /etc/rc.d/ o /etc/init.d/.
* Estos scripts inicializan servicios del sistema (demonios de red, syslog, cron, etc.).
* Se usan enlaces simbólicos en directorios como /etc/rc2.d/, /etc/rc3.d/, etc., que apuntan a los scripts en /etc/init.d/.
* Los scripts comienzan con S (Start) o K (Kill), indicando qué servicios iniciar o detener en ese runlevel.

**6) Configuración del entorno multiusuario**

* Se montan sistemas de archivos definidos en /etc/fstab.
* Se configuran interfaces de red, swap, demonios y servicios de usuario.

**7) Inicialización de terminales y login**

* init lanza procesos getty en las terminales (consolas virtuales o TTYs).
* getty muestra el prompt de login.
* Al introducir usuario y contraseña, se ejecuta login, que valida credenciales y abre la shell del usuario.

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

init es el primer proceso de espacio de usuario, ejecutado por el kernel, y su objetivo es poner en marcha el sistema en el runlevel configurado, lanzar servicios, y garantizar que el usuario pueda loguearse e interactuar con el sistema.

**c. RunLevels. ¿Qué son? ¿Cuál es su objetivo?**

Los runlevels son modos de ejecución predeterminados del sistema en System V, cuyo objetivo es definir qué servicios y procesos deben estar activos en cada estado del sistema, desde apagado hasta ejecución en modo gráfico.

**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**: modo monousuario, solo root, sin red; usado para tareas de mantenimiento.
* **2: multiuser, without NFS** (modo multiusuario sin soporte de red): varios usuarios pero sin servicios de red.
* **3: full multiuser mode console** (modo multiusuario completo por consola): varios usuarios, servicios de red activos, pero **solo consola de texto**.
* **4: No definido (custom)** → reservado para usos especiales o personalizados.
* 5: **Multi-user con interfaz gráfica** → similar al runlevel 3 pero inicia además el servidor gráfico (X11, display manager).
* 6: **Reboot** → reinicia el sistema.

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.

**¿Todas las distribuciones respetan estos estándares?**

**No.** Si bien los runlevels provienen del estándar System V, no todas las distribuciones los implementan igual:

* En **Debian/Ubuntu**: el runlevel **2** ya se usa como multiusuario con red (no como "sin red" según el estándar).
* En **Red Hat/CentOS/Fedora**: siguen más fielmente la definición clásica (3 = multiusuario en texto, 5 = multiusuario gráfico).
* En distribuciones modernas que usan **systemd** (la mayoría actuales), los runlevels fueron reemplazados por **targets** (graphical.target, multi-user.target, etc.), aunque se mantienen comandos de compatibilidad (systemctl isolate runlevel3.target).

**e. Archivo /etc/inittab. ¿Cuál es su ﬁnalidad? ¿Qué tipo de información se almacena en el? ¿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) deberá ejecutar 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 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**](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 isocso a su instalación de GNU/Linux, especiﬁque que su home sea creada en /home/isocso, y hágalo miembro del grupo informatica (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

**1. Crear el grupo informatica (si no existe)**

Primero asegurarse de que el grupo esté creado:

* **sudo groupadd informática**

Si ya existía, te va a dar un error, pero no pasa nada.

**2. Crear el usuario isocso con home en /home/isocso y miembro de informática**

* **sudo useradd -m -d /home/isocso -g informatica isocso**

Explicación:

* -m → crea el directorio home.
* -d /home/isocso → define la ruta del home.
* -g informatica → asigna el grupo primario

Le podés poner contraseña:

* sudo passwd isocso

**3. Crear un archivo en su home sin loguearte como ese usuario**

Podés hacerlo como root:

* **sudo touch /home/isocso/archivo.txt**
* **sudo chown isocso:informatica /home/isocso/archivo.txt**

Con eso, el archivo pertenece al usuario y al grupo correctos**.**

Verificación:

* **sudo ls -l /home/isocso/**

Deberías ver algo como:

* **-rw-r--r-- 1 isocso informatica 0 sep 2 20:46 /home/isocso/archivo.txt**

**4. Borrar al usuario y su home**

* **sudo userdel -r isocso**

El -r asegura que también se borre el directorio /home/isocso.

**5. Verificar que no queden registros**

Los usuarios están en /etc/passwd, y los grupos en /etc/group.

Busca al usuario:

* **grep isocso /etc/passwd**

Busca en grupos:

* **grep isocso /etc/group**

Si no aparece nada → ya no queda rastro.

**16. FileSystem y permisos:**

**(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):

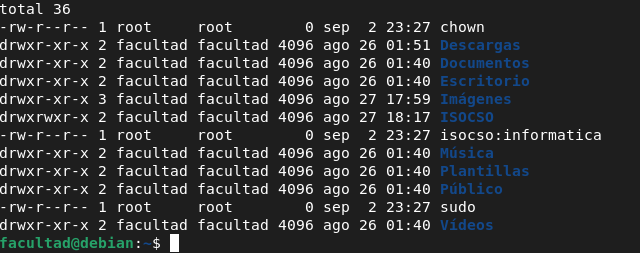
* En archivos: Pueden leerse los contenidos del archivo.
* En directorios: Permite detallar los contenidos del directorio.

- 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.



**Ejemplo de la captura**

Tomemos esta línea de tu salida:

* drwxr-xr-x 2 facultad facultad 4096 ago 26 01:40 Documentos

Se interpreta así:

* d → es un directorio.
* rwx → el **dueño** (facultad) puede leer, escribir y entrar.
* r-x → el **grupo** (facultad) puede leer y entrar, pero no escribir.
* r-x → los **otros** usuarios pueden leer y entrar, pero no escribir.
* 2 indica el numero de enlaces al archivo o directorio
* “facultad” indica el usuario propietario
* “facultad” indica el grupo al que pertenece el archivo o directorio
* 4096 indica en peso en bytes
* Ago 26 01:40 indica hora y fecha de la ultima modificación
* Documentos indica el nombre del directorio.

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

* **Chmod**
* **chown**
* **hgrp**

**1. chmod (change mode)**

* **Funcionalidad:** Permite cambiar los permisos de acceso (lectura, escritura y ejecución) de un archivo o directorio. Se pueden definir permisos para el **dueño (u)**, el **grupo (g)** y los **otros usuarios (o)**.
* **Parámetros más comunes:**
  + -R → aplica los cambios de forma recursiva en subdirectorios.
  + Permisos en modo **simbólico** (u+r, g-w, o=x) o **numérico (octal)** (755, 644).

**2. chown (change owner)**

* **Funcionalidad:** Cambia el **propietario (usuario)** de un archivo o directorio y, opcionalmente, su grupo asociado.
* **Parámetros más comunes:**
  + -R → aplicar cambios de forma recursiva.
  + --reference=archivo → asigna propietario y grupo iguales a otro archivo.
* **Sintaxis típica:**
  + chown usuario archivo
  + chown usuario:grupo archivo

**3. chgrp (change group)**

* **Funcionalidad:** Cambia el **grupo** asociado a un archivo o directorio, sin modificar al propietario.
* **Parámetros más comunes:**
  + -R → aplicar cambios de forma recursiva.
  + --reference=archivo → asigna el mismo grupo que tiene otro archivo.
* **Sintaxis típica:**
  + chgrp grupo archivo

**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?**

Cuando se utiliza chmod, además de la notación simbólica (u+r, g-w, etc.), se puede usar la notación octal.

Esta notación representa los permisos de usuario (dueño), grupo y otros mediante un número de tres cifras en base 8 (octal). Cada permiso se traduce en un valor:

* r (read) → 4
* w (write) → 2
* x (execute) → 1

Los valores se suman para obtener el número de cada categoría.

**Ejemplos**

* 7 = 4 + 2 + 1 = **rwx** (lectura, escritura y ejecución).
* 6 = 4 + 2 = **rw-** (lectura y escritura, sin ejecución).
* 5 = 4 + 1 = **r-x** (lectura y ejecución, sin escritura).
* 4 = solo **r--** (lectura).
* 0 = sin permisos.

**Cómo se interpretan las tres cifras**

El número completo tiene **tres dígitos**, cada uno asignado a:

1. **Primer dígito** → permisos del **dueño (u)**.
2. **Segundo dígito** → permisos del **grupo (g)**.
3. **Tercer dígito** → permisos de **otros (o)**.

chmod 755 script.sh

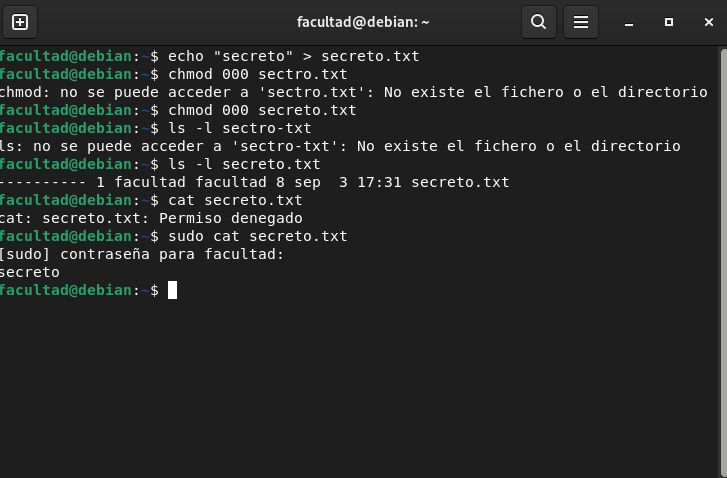
7 = rwx → dueño puede todo.  
5 = r-x → grupo puede leer y ejecutar, no escribir.  
5 = r-x → otros pueden leer y ejecutar, no escribir.

Resultado: rwxr-xr-x.

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

En condiciones normales, **no**. El kernel controla los permisos definidos en el sistema de archivos, si un usuario no tiene permisos de lectura/escritura/ejecución sobre un archivo o directorio → no puede acceder.

👉 **Pero sí existen excepciones:**

1. **El superusuario (root):** Root puede leer, escribir o ejecutar cualquier archivo, sin importar los permisos. Ejemplo: aunque un archivo tenga permisos ---------, root lo puede abrir.
2. **Usuarios con permisos elevados (sudo):** Si un usuario puede ejecutar sudo, puede actuar como root y, por lo tanto, acceder a cualquier archivo.
3. **Capacidades especiales o atributos extendidos:** Archivos con *SUID*, *SGID* o *ACLs* pueden dar permisos adicionales. Ejemplo: /bin/passwd tiene el bit SUID para que cualquier usuario pueda cambiar su contraseña.

**e) Explique los conceptos de “full path name” (path absoluto) y “relative path name” (path relativo). De ejemplos claros de cada uno de ellos.**

**📂 Full path name (path absoluto)**

* Es la **ruta completa** de un archivo o directorio, comenzando desde la raíz del sistema (/).
* No depende del directorio en el que se encuentre el usuario.
* Siempre empieza con /.

**Ejemplos:**

* /home/facultad/documentos/tp1.txt
* /etc/passwd
* /usr/bin/ls

👉 Si escribís esa ruta en cualquier lugar del sistema, siempre apunta al mismo archivo.

**📂 Relative path name (path relativo)**

* Es la ruta que se indica **en relación al directorio actual** en el que está el usuario (el *working directory*).
* No comienza con /.
* Usa notaciones como:
  + . → el directorio actual.
  + .. → el directorio padre.

**Ejemplos (suponiendo que estamos en /home/facultad):**

* documentos/tp1.txt → equivale a /home/facultad/documentos/tp1.txt.
* ./documentos/tp1.txt → lo mismo, pero indicando explícitamente “desde aquí”.
* ../otro\_usuario/archivo.txt → va un nivel arriba y entra a la carpeta otro\_usuario.

**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) Comandos relacionados con el uso del FileSystem en GNU/Linux**

**1. cd (*change directory*)**

* **Funcionalidad:** Cambia el directorio de trabajo actual.
* **Parámetros comunes:**
  + cd /ruta/absoluta → ir a una ruta absoluta.
  + cd .. → subir un nivel.
  + cd ~ o cd → ir al directorio personal.
  + cd - → volver al directorio anterior.

**2. umount (*unmount*)**

* **Funcionalidad: Desmonta un sistema de archivos montado.**
* **Parámetros comunes:**
  + umount /ruta/punto\_montaje → desmontar por punto de montaje.
  + umount /dev/sdX → desmontar por dispositivo.

**3. mkdir (*make directory*)**

* **Funcionalidad: Crea directorios.**
* **Parámetros comunes:**
  + mkdir nombre → crea un directorio.
  + mkdir -p ruta/larga/ejemplo → crea jerarquía de directorios, aunque no existan los intermedios.

**4. du (*disk usage*)**

* **Funcionalidad:** Muestra el espacio en disco ocupado por archivos o directorios.
* **Parámetros comunes:**
  + du -h → muestra tamaños en formato legible (KB, MB, GB).
  + du -s → muestra solo el total.
  + du -sh \* → tamaño de cada archivo/directorio en el directorio actual.

**5. rmdir (*remove directory*)**

* **Funcionalidad:** Elimina directorios vacíos.
* **Parámetros comunes:**
  + rmdir nombre → elimina un directorio vacío.
  + rmdir -p ruta/larga → elimina directorios en cascada si están vacíos.

**6. df (*disk free*)**

* **Funcionalidad:** Muestra el espacio disponible y usado en los sistemas de archivos **montados.**
* **Parámetros comunes:**
  + df -h → formato legible.
  + df -T → muestra tipo de sistema de archivos.

**7. mount**

* **Funcionalidad:** Monta un dispositivo o partición en un directorio del sistema.
* **Parámetros comunes:**
  + mount /dev/sdX /mnt/punto → monta un dispositivo en un punto de montaje.
  + mount -t tipo → especifica el tipo de sistema de archivos (ext4, vfat, etc.).

**8. ln (*link*)**

* **Funcionalidad:** Crea enlaces entre archivos.
* **Parámetros comunes:**
  + ln archivo enlace → enlace **duro**.
  + ln -s archivo enlace → enlace **simbólico (soft link)**.

**9. ls (*list*)**

* **Funcionalidad:** Lista archivos y directorios.
* **Parámetros comunes:**
  + ls -l → listado largo con permisos, dueños y fechas.
  + ls -a → incluye archivos ocultos.
  + ls -lh → formato legible (KB, MB).
  + ls -R → recursivo.

**10. pwd (*print working directory*)**

* **Funcionalidad:** Muestra la ruta absoluta del directorio actual.
* **Parámetros comunes:** no suele requerir parámetros.

**11. cp (*copy*)**

* **Funcionalidad:** Copia archivos o directorios.
* **Parámetros comunes:**
  + cp archivo destino/ → copia archivo.
  + cp -r directorio destino/ → copia recursiva (para directorios).
  + cp -i → pide confirmación antes de sobrescribir.

**12. mv (*move*)**

* **Funcionalidad:** Mueve o renombra archivos y directorios.
* **Parámetros comunes:**
  + mv archivo destino/ → mueve archivo.
  + mv archivo nuevo\_nombre → renombra archivo.
  + mv -i → pide confirmación antes de sobrescribir.

✅ **Resumen corto para examen:**

* cd → cambia directorio.
* umount/mount → desmonta/monta sistemas de archivos.
* mkdir/rmdir → crea o elimina directorios.
* ls/pwd → lista archivos y muestra directorio actual.
* cp/mv → copia o mueve/renombra archivos.
* du/df → muestran uso de espacio en archivos o discos.
* ln → crea enlaces (duros o simbólicos).

**17. Procesos:**

**a. ¿Qué signiﬁca que un proceso se está ejecutando en Background? ¿Y en Foreground?**

**b. ¿Cómo puedo hacer para ejecutar un proceso en Background? ¿Como puedo hacer para pasar un proceso de background a foreground y viceversa?**

**🔹 Foreground (primer plano)**

* Un proceso en foreground es aquel que se ejecuta ocupando la terminal actual.
* Mientras está corriendo, la terminal queda bloqueada para otros comandos hasta que el proceso finaliza.

Ejemplo: cat archivo.txt

Mientras cat muestra el contenido, no podés escribir otro comando en esa terminal.

**🔹 Background (segundo plano)**

* Un proceso en background se ejecuta independientemente de la terminal.
* Permite al usuario seguir usando la misma terminal para otros comandos mientras el proceso sigue corriendo.
* Para lanzarlo en segundo plano se agrega & al final: **gedit archivo.txt &**

**Aquí gedit se abre y la terminal queda libre para seguir trabajando.**

**🔹 Relación práctica**

* **Foreground:**
  + Bloquea la terminal hasta terminar.
  + Se puede “mandar” al background con Ctrl+Z y luego bg.
* **Background:**
  + La terminal queda disponible.
  + Si cerras la terminal, el proceso suele terminar (a menos que uses nohup o screen).
  + Podes traerlo al foreground con fg.

**c. 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.

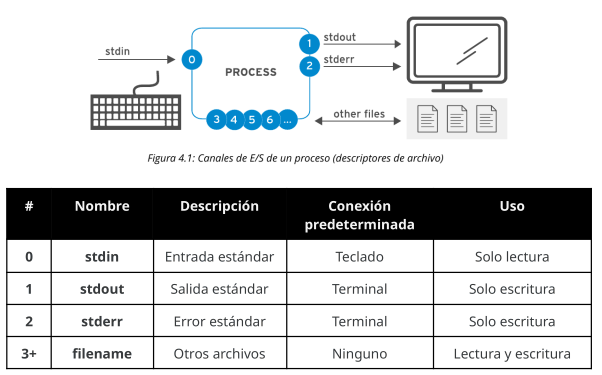
**d. Redirección. ¿Qué tipo de redirecciones existen? ¿Cuál es su ﬁnalidad? Cite ejemplos de utilización.**

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.

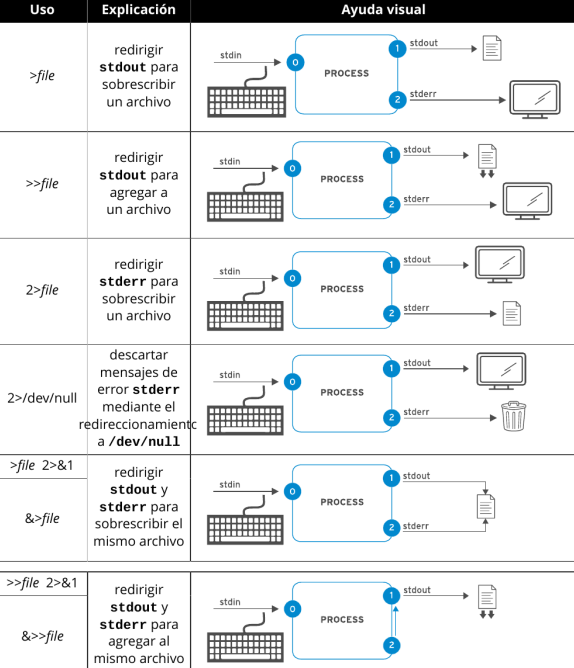
* Entrada estándar (canal 0) lee entradas desde el teclado.
* Salida estándar (canal 1) envia una salida normal al terminal.
* Error estándar (canal 2) envia mensajes de error al terminal.

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

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.

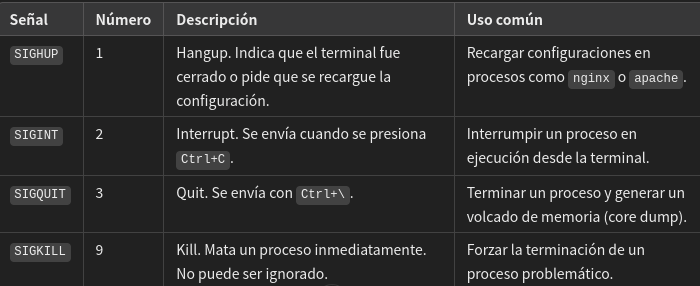
El redireccionamiento de stdout evita que la salida de un proceso aparezca en el terminal. Como se puede ver en la siguiente tabla, el redireccionamiento de únicamente stdout no evita que los mensajes de error stderr aparezcan en el terminal. Si el archivo no existe, se creará. Si el archivo existe y 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.

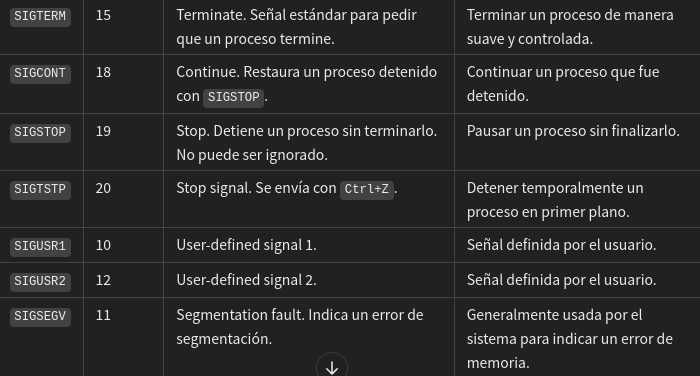


**g) Comando kill. ¿Cuál es su funcionalidad? Cite ejemplos**

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:





Ejemplos:

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

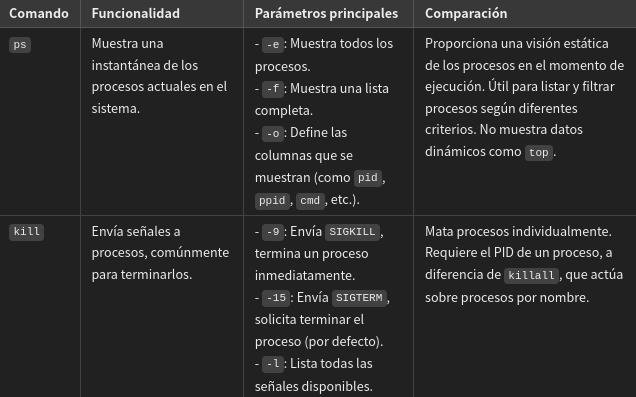
- kill -9 1234 # Envía SIGKILL al proceso con PID 1234.

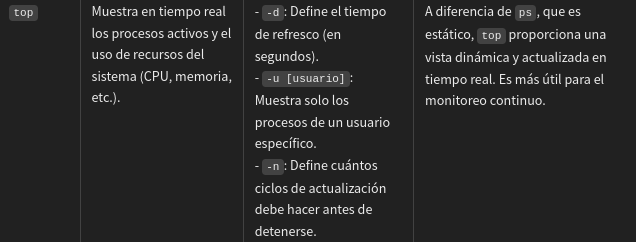
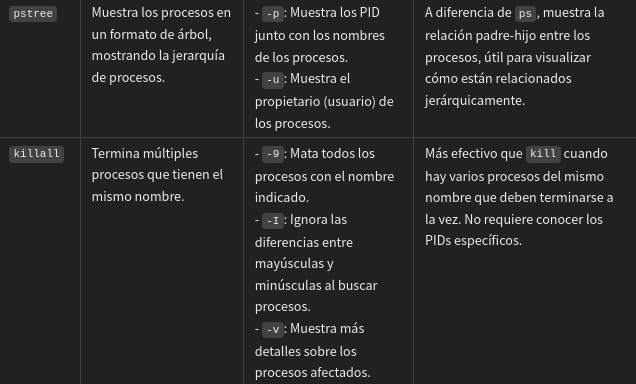
- kill -HUP 1234 # Envía SIGHUP al proceso con PID 1234.

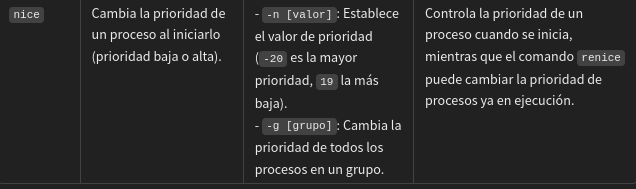
- kill -u usuario # Envía SIGTERM a todos los procesos del usuario.

**(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







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