PRACTICA 1

¿Qué es GNU?

Es un Sistema Operativo tipo Unix (Unix like), pero libre.

* S.O. diseñado por miles de programadores
* S.O. gratuito y de libre distribución (se baja desde la Web, CD, etc.)
* Existen diversas distribuciones (customizaciones)
* Es código abierto, lo que nos permite estudiarlo, personalizarlo, auditarlo, aprovecharnos de la documentación, etc...
* Permite a los usuarios tener la libertad de usar, estudiar, distribuir y mejorar los programas

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

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

*Breve historia*

*Fue iniciado por Richard Stallman en 1983 con el fin de crear un Unix libre (el sistema GNU). En 1985 Richard Stallman creó la Free Software Foundation (FSF o Fundación para el Software Libre) para proveer soportes logísticos, legales y financieros al proyecto GNU. Esta fundación contrato programadores, aunque una porción sustancial del desarrollo fue (y continúa siendo) producida por voluntarios.*

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

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

Características más importantes de GNU/Linux:

* Es multiusuario
* Es multitarea y multiprocesador
* Es altamente portable
* Posee diversos intérpretes de comandos, de los cuales algunos son programables
* Permite el manejo de usuarios y permisos
* Todo es un archivo (hasta los dispositivos y directorios)
* Cada directorio puede estar en una partición diferente (/tmp, /home, etc.)
* Es case sensitive
* Es código abierto

¿Qué es POSIX?

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

Es un conjunto de estándares que define varias interfaces de herramientas, comandos y API para Sistemas Operativos similares a UNIX y otros; una norma escrita por la IEEE, que define una interfaz estándar del sistema operativo y el entorno, incluyendo un intérprete de comandos (shell)

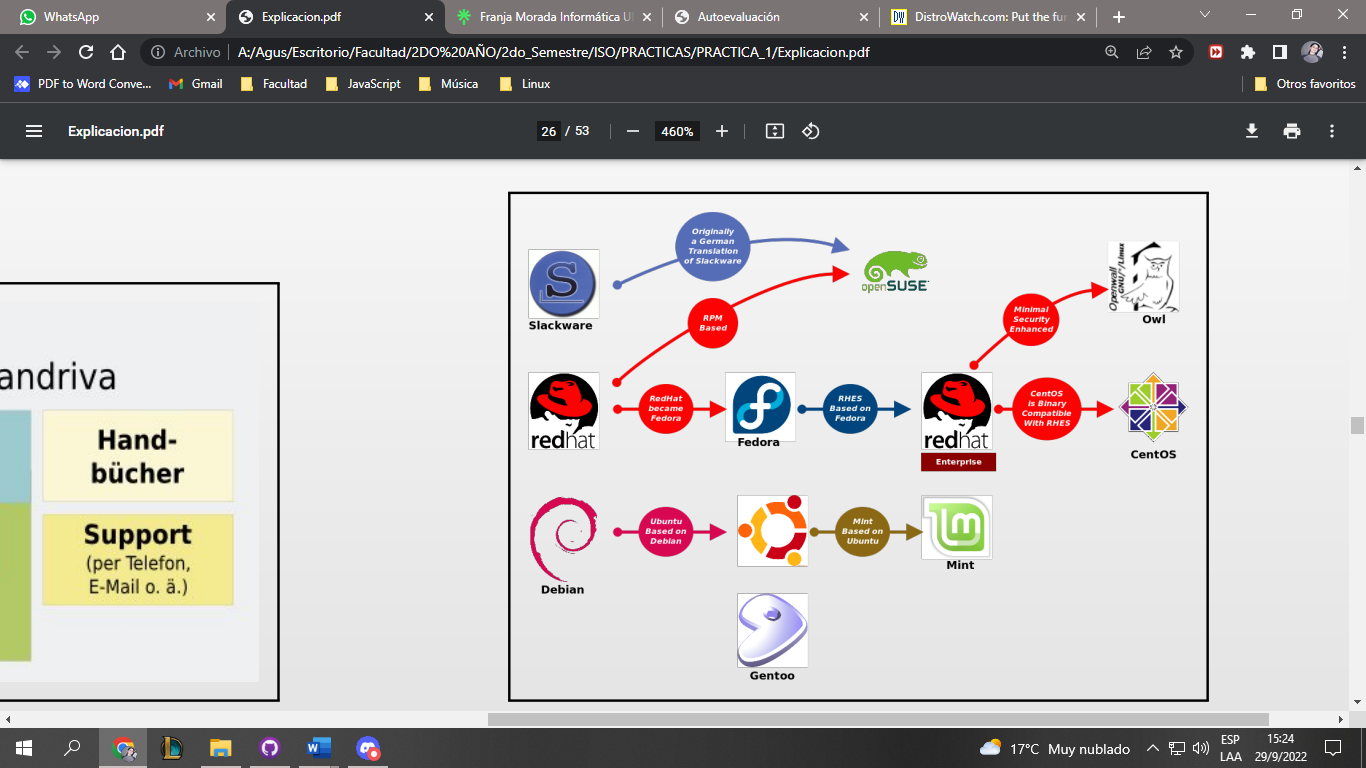
Distribuciones

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

Una distribución es una customización de GNU/Linux formada por una versión de kernel y determinados programas con sus configuraciones

Existen distribuciones que incluyen paquetes propietarios y otras que son 100% libres.

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



Estructura GNU/Linux

3 componentes fundamentales de GNU/Linux.

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

**La estructura básica del S.O es el Kernel que:**

* Ejecuta programas y gestiona dispositivos de hardware
* Es el encargado de que el software y el hardware puedan trabajar juntos
* Sus funciones más importantes son la administración de memoria, CPU y la E/S
* En sí, y en un sentido estricto, es el sistema operativo
* Es un núcleo monolítico híbrido:
  + Los drivers y código del Kernel se ejecutan en modo privilegiado
  + Lo que lo hace híbrido es la capacidad de cargar y descargar funcionalidad a través de módulos.
* Esta licenciado bajo la licencia GPL v2

**Kernel**

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

La versión estable actual (según la practica) es 5.8.X.

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.

Nomenclatura versionado:

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

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

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

**¿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

**Shell**

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.

* También conocido como CLI (Command Line Interface).
* Modo de comunicación entre el usuario y el SO.
* Ejecuta programas a partir del ingreso de comandos.
* Cada usuario puede tener una interfaz o Shell. Esta se define por defecto (si no se especifica) pero puede ser personzalida.
  + Un usuario puede cambiar su propio shell a cualquier cosa: la cual, sin embargo, debe estar listada en el archivo/etc/shells.
  + Solo root puede ejecutar un shell que no esté incluido en el archivo/etc/shells.
  + Si una cuenta tiene un shell de inicio de sesión restringido, solo root puede cambiar el shell de ese usuario.
  + Al crear cuentas de usuario con las utilidades useradd o adduser, la marca --shell se puede usar para especificar el nombre del shell de inicio de sesión de un usuario que no sea el especificado en los archivos de configuración respectivos.
* Se pueden personalizar.
* Son programables.
* Bourne Shell (sh), Korn Shell (ksh), Bourne Again Shell (bash)(autocompletado, history, alias).

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.

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

El shell nos permite ejecutar:

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

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

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

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

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

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

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

Los comandos internos son nativos del shell de linux que estemos usando (bash por ejemplo). Estos se suelen encontrar en el directorio /usr/bin. Los externos se encuentran en la variables $PATH, no son nativos del shell de Linux.

**FileSystem**

El Filesystem 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.

El adoptado por GNU/Linux es el Extended (v2, v3, v4).

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.

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. Estan los comandos que pueden usar todos los usuarios (incluido el root).

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

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

/etc: Contiene la configuración de los paquetes instalados.

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

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

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

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

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

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

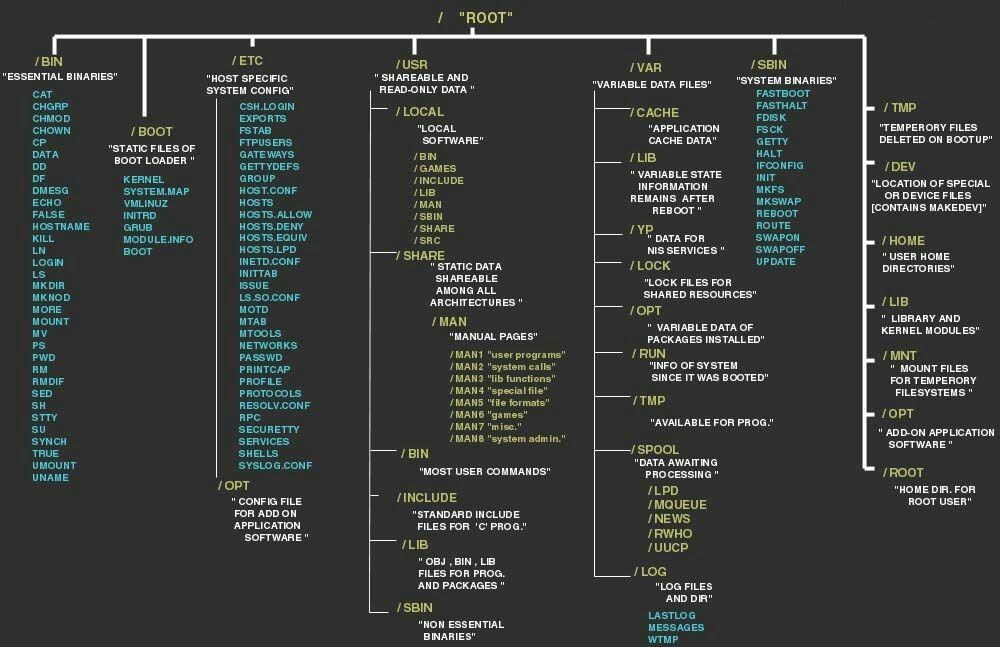
/root: La carpeta home para el usuario root.

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

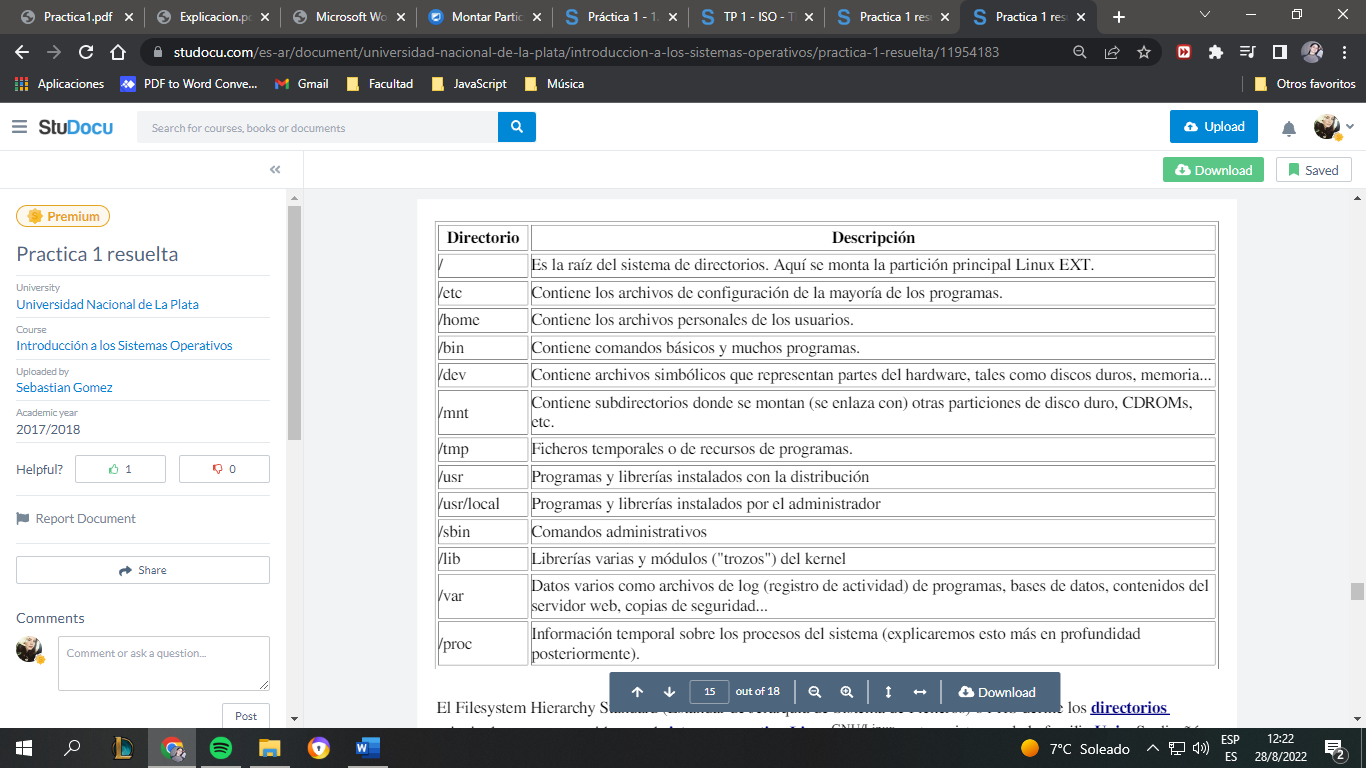
/tmp: Contiene los archivos temporales.

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

/var: Contiene registros del sistema y otros datos variables



**ANOTACION: /dev/null es un archivo especial que descarta toda la información que se escribe en o se redirige hacia él. A su vez, no proporciona ningún dato a cualquier proceso que intente leer de él, devolviendo simplemente un EOF o fin de archivo.**



PARTICIONES

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

Ventajas y deventajas:

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

-Tener una partición de Restore de todo el sistema

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

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

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.

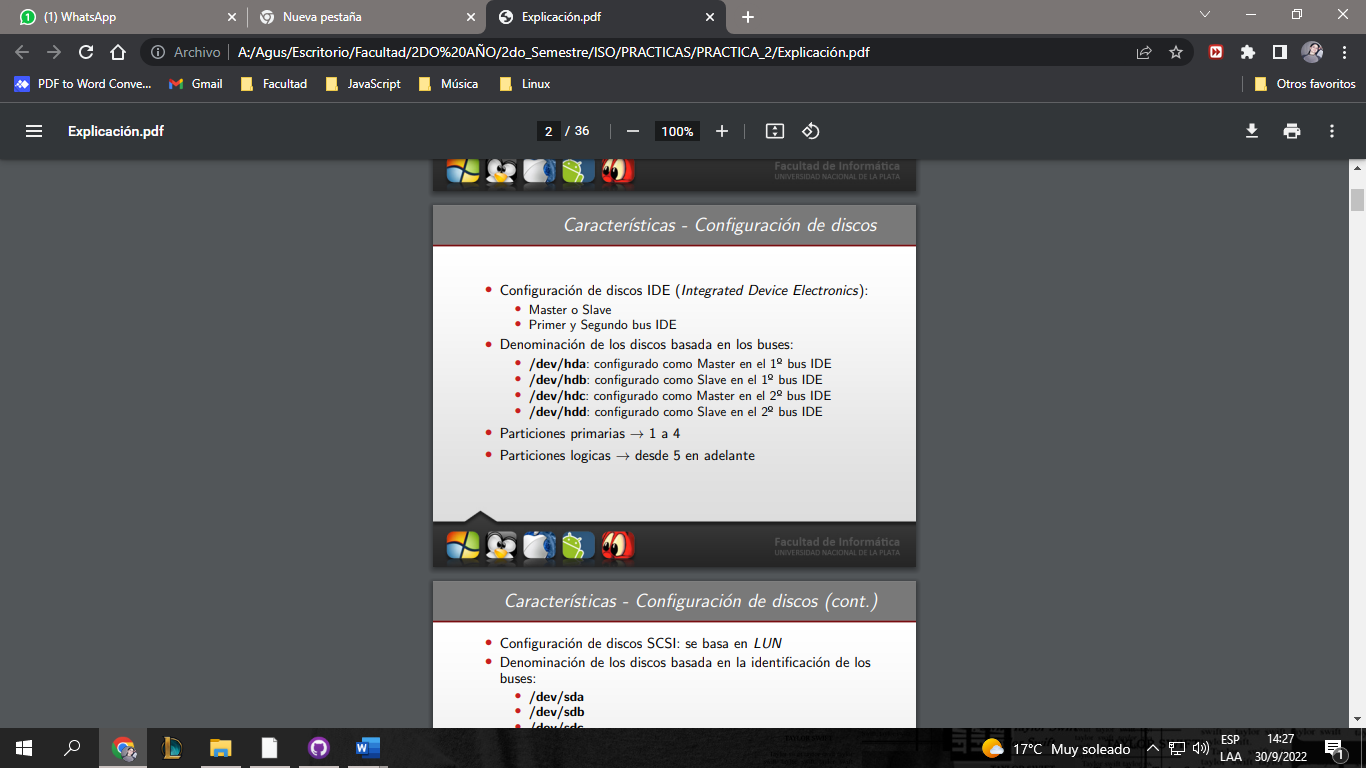
Partición extendida: sirve para contener unidades lógicas en su interior. Solo puede existir una partición de este tipo por disco. No se define un tipo de FS directamente sobre ella.

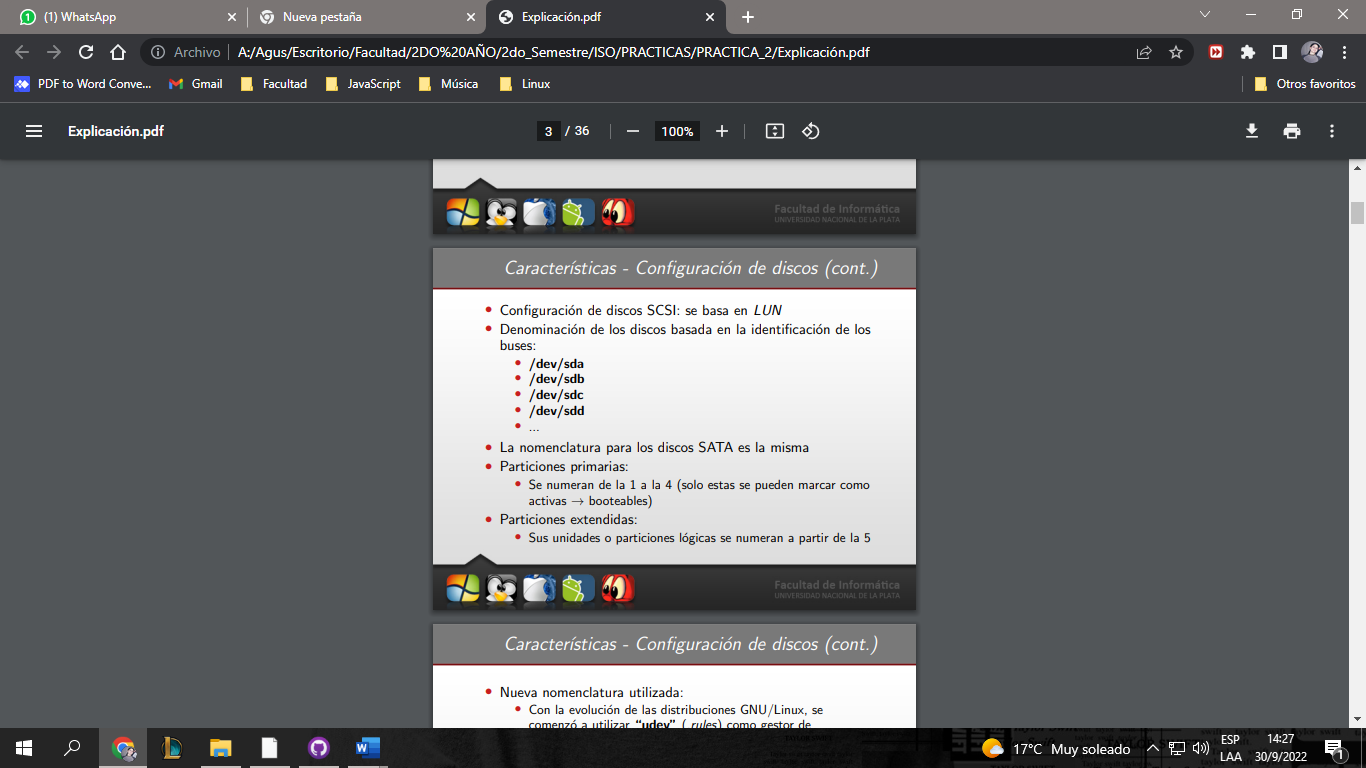
Partición lógica: ocupa la totalidad o parte de la partición extendida y se le define un tipo de FS. Las particiones de este tipo se conectan como una lista enlazada.

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

* Disqueteras
  + Primera disquetera: **/dev/fd0** (en Windows sería la disquetera A:)
  + Segunda disquetera: **/dev/fd1**
* Discos duros (en general: **/dev/hdx#,** donde x es el disco y # es la partición)
  + Primer disco duro: (todo el disco) **/dev/hda**
    - Particiones primarias
      * Primera partición primaria: **/dev/hda1**
      * Segunda partición primaria: **/dev/hda2**
      * Tercera partición primaria: **/dev/hda3**
      * Cuarta partición primaria: **/dev/hda4**
    - Particiones lógicas
      * Primera partición lógica: **/dev/hda5**
      * Sucesivamente: **/dev/hda#**
  + Segundo disco duro: (todo el disco**) /dev/hdb**
    - Particiones primarias
      * Primera partición primaria: **/dev/hdb1**
      * Segunda partición primaria: **/dev/hdb2**
      * Tercera partición primaria: **/dev/hdb3**
      * Cuarta partición primaria: **/dev/hdb4**
    - Particiones lógicas
      * Primera partición lógica: **/dev/hdb5**
      * Sucesivamente: **/dev/hdb#**
* Discos SCSI
  + Primer disco SCSI: **/dev/sda**
  + Segundo disco SCSI: **/dev/sdb**
    - Sucesivamente ...

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







Para instalar GNU/Linux 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.

Si no has separado /home del resto del sistema, cuando vayas a instalar una nueva versión, perderás toda esa documentación.

Punto de montaje / 🡪 Primaria. Sistema de ficheros ext4 transaccional.

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

Existen 2 tipos:

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

Arranque (bootstrap) de un Sistema Operativo:

En las arquitecturas x86 la BIOS de la motherboardes un chip especial que guarda configuración inicial de la computadora.

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.

**EFI**

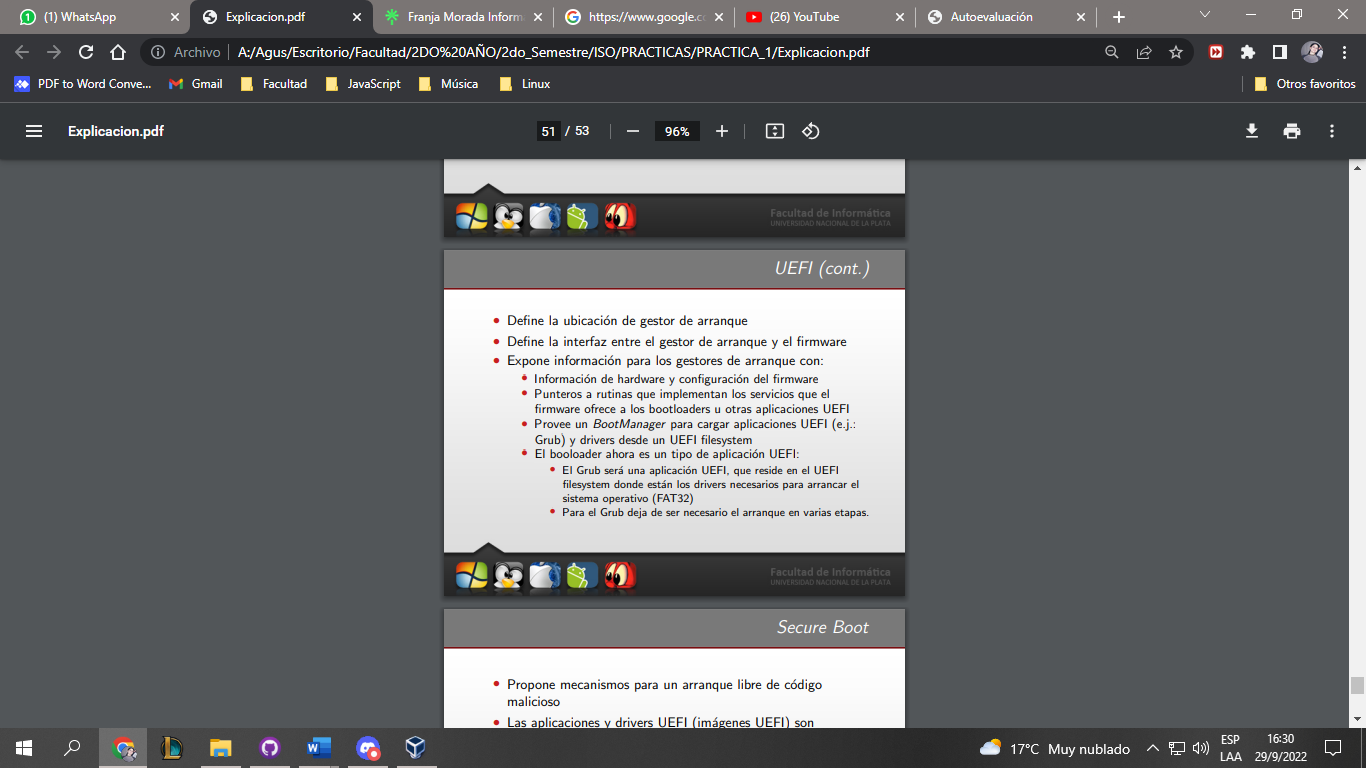
**EFI** es nexo entre el SO y el firmware:

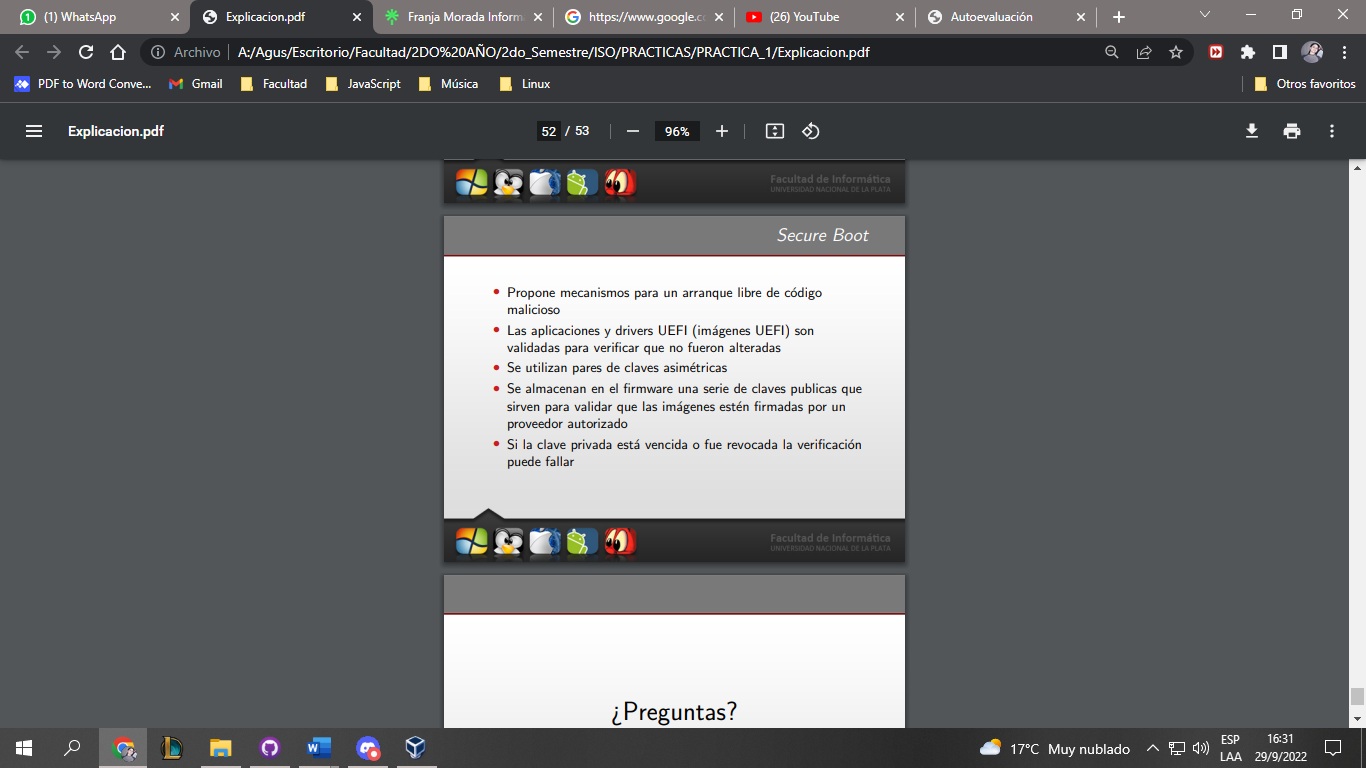
* Utiliza el sistema GPT (GUID partition table) para solucionar limitaciones del MBR, como la cantidad de particiones.
* GPT especifica la ubicación y formato de la tabla de particiones en un disco duro.
* Es parte de EFI. Puede verse como una sustitución del MBR.
* La especificación EFI es propiedad de Intel
* Alternativa para reemplazar la BIOS
* Se pueden crear hasta 128 particiones
* Se mantiene un MBR para tener compatibilidad con el esquema BIOS
* GPT usa un modo de direccionamiento lógico (logical block addressing LBA) en lugar de cylinder-header-sector. GPT puede verse como una sustitución del MBR.
* El MBR “heredado” se almacena en el LBA 0.
* En el LBA 1 está la cabecera GPT. La tabla de particiones en sí está en los bloques sucesivos
* La cabecera GPT y la tabla de particiones están escritas al principio y al final del disco (redundancia)

**UEFI**

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

* Alianza entre varias compañías con el objetivo de modernizar el proceso de arranque.
* UEFI es propiedad del UEFI Forum.
* Define la interfaz entre el gestor de arranque y el firmware más sencilla y fácil de comprender.
* Aporta mayor velocidad en el arranque de los equipos
* UEFI mejora la seguridad con su funcionalidad Secure Boot

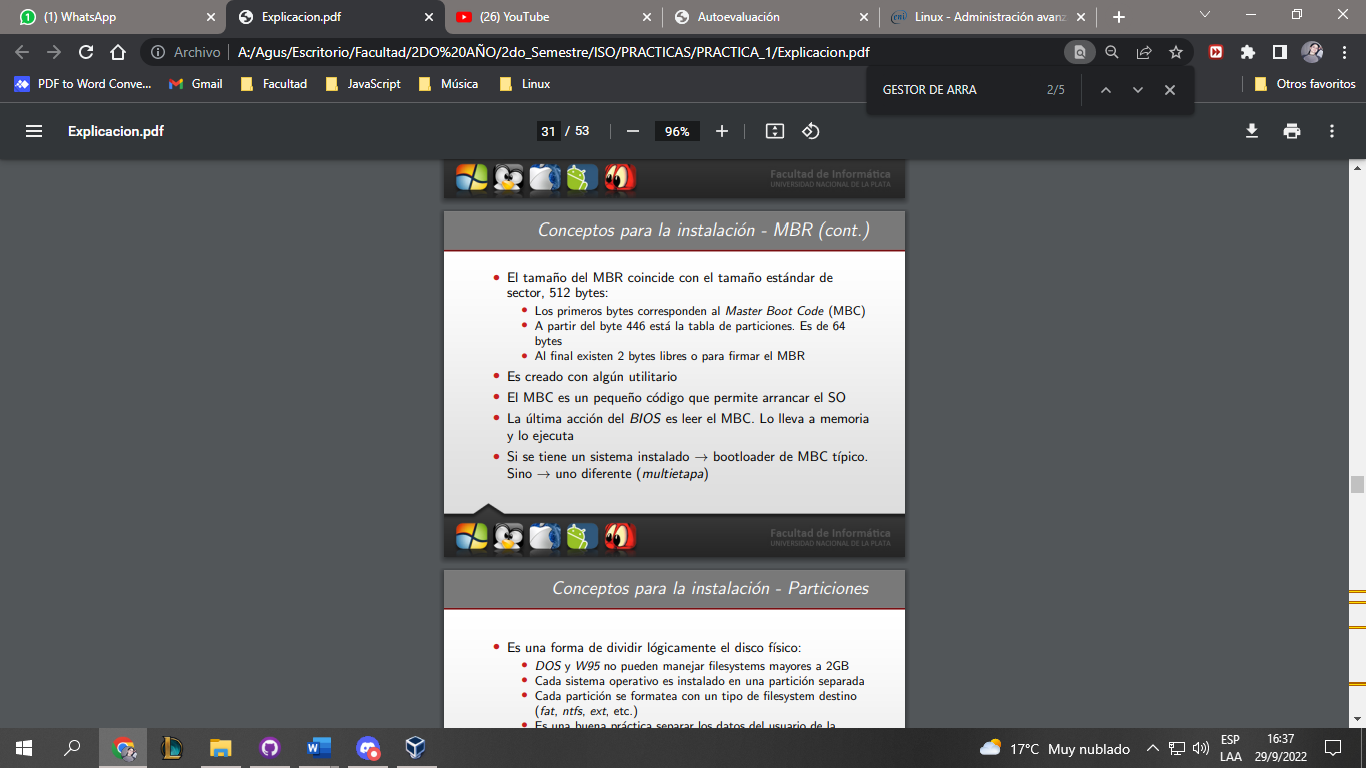




**El MBR** (master boot record) es el primer sector del disco (cilindro 0, cabeza 0, sector 1). Esto se carga a memoria y se ejecuta. Es un registro de arranque principal que puede contener un código de arranque denominado MBC (master boot code) y una marca de 2 bytes que indica su presencia o puede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

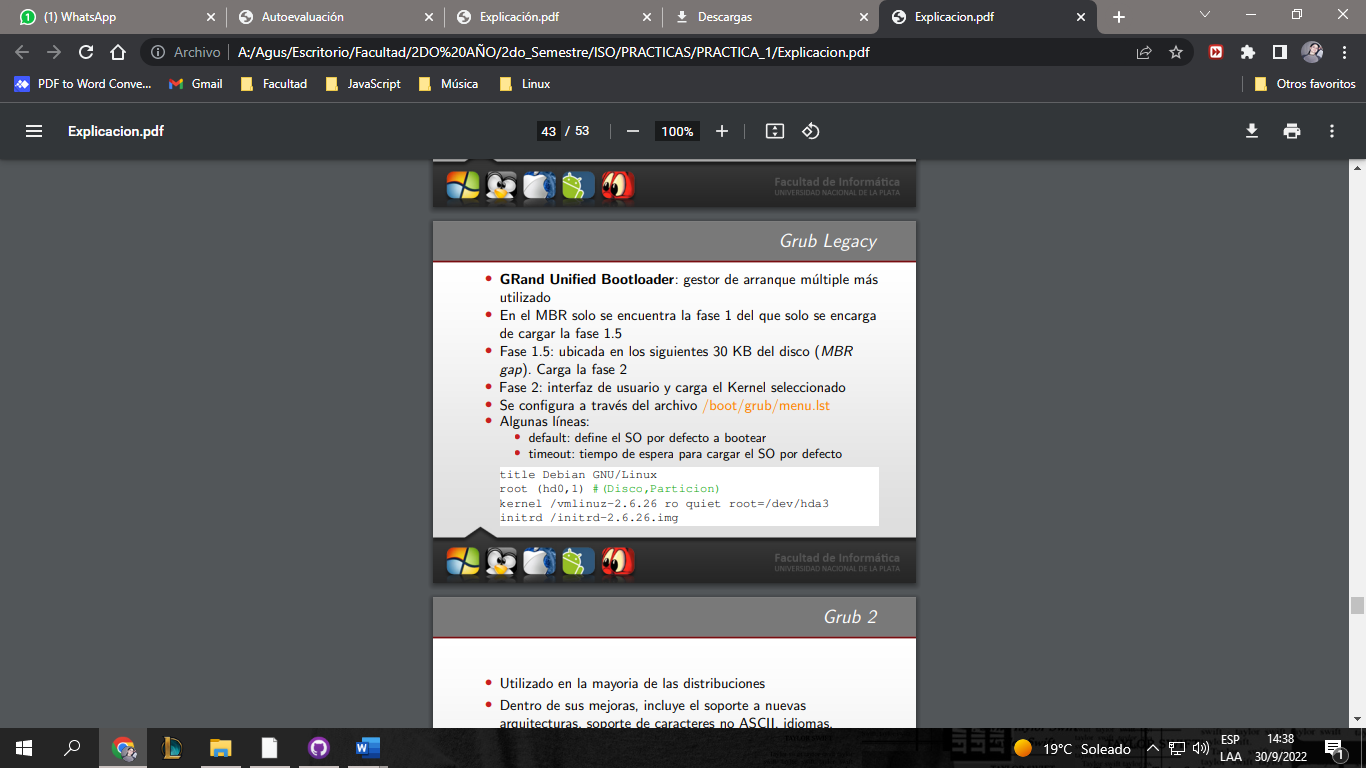
Existe un MBR en todos los discos y si existiese más de un disco rígido en la máquina, sólo uno es designado como Primary Master Disk. El tamaño del MBR coincide con el tamaño estándar de sector: 512 bytes.

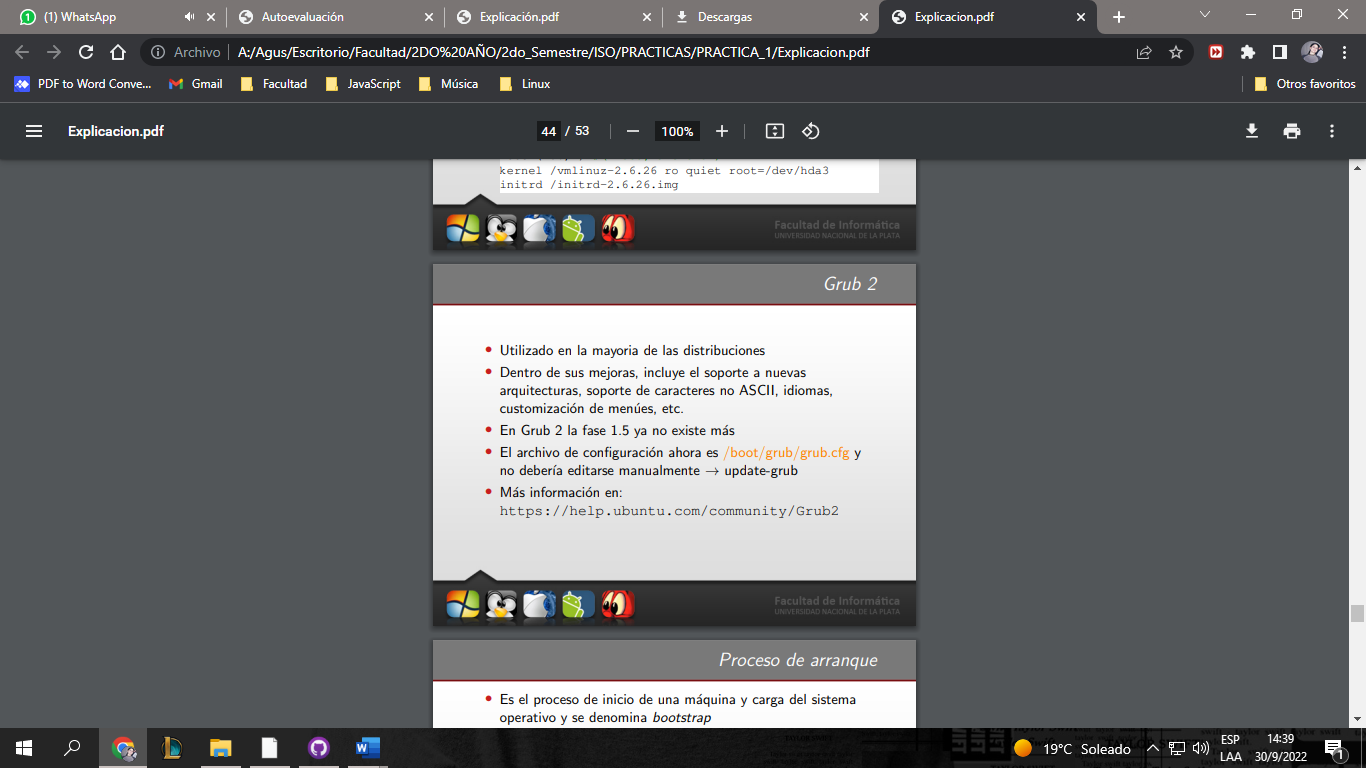
El MBC es un pequeño código que permite arrancar el SO. La última acción del BIOS es leer el MBC, lo lleva a memoria y lo ejecuta.



El bootloader o cargador de arranque es un programa que permite cargar el Sistema Operativo. Puede llegar a cargar un entorno previo a la carga del sistema

* Generalmente se utilizan los cargadores multietapas, en los que varios programas pequeños se van invocando hasta lograr la carga del SO
* En cierto sentido, el código del BIOS/UEFI forma parte del bootloader, pero el concepto esta más orientado al código que reside en el Master Boot Record (512b)
* El MBR esta formado por el MBC (446b) y la Tabla de Particiones (64b)
* Solo el MBC del Primary Master Disk es tenido en cuenta
* El MBR existe en todos los discos, ya que contiene la tabla de particiones





Entonces:

- 1. Se ejecuta el código de la BIOS

- 2. El hardware lee el sector de arranque

- 3. Se carga el gestor de arranque

- 4. Se carga el kernel

**¿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.

Archivos

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

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

Comandos

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

More: Al igual que ‘cat’, ‘more’ permite visualizar por pantalla el contenido de un fichero de texto, con la diferencia con el anterior de que ‘more’ página 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:

**(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]

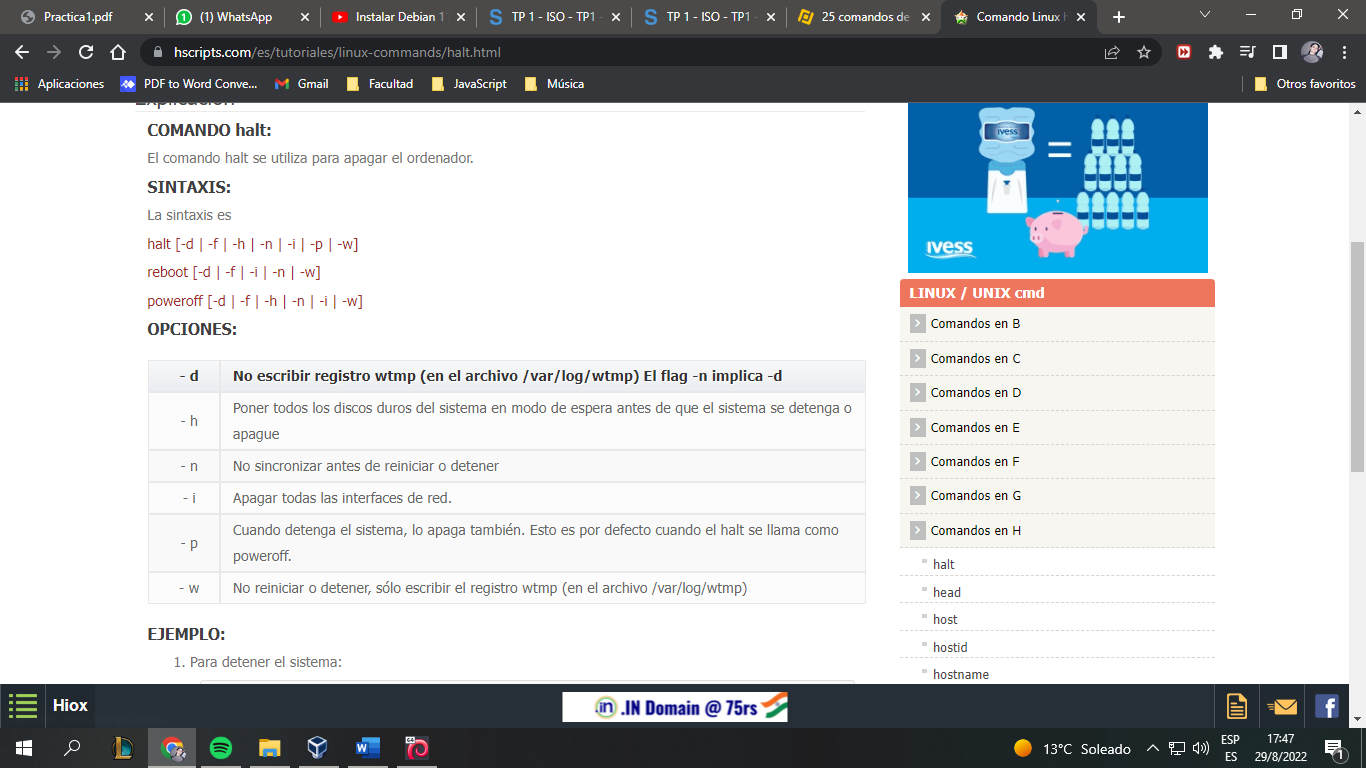
**(b) reboot**

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/>

EMULADORES/VIRTUALIZADORES

* Básicamente se pueden considerar 3 tipos:
  + Emulación:
    - Emulan hardware
    - Tienen que implementar todas las instrucciones de la CPU
    - Es muy costosa y poco eficiente
    - Permite ejecutar arquitecturas diferentes a las soportadas por el hardware
* Virtualización completa:
  + Permiten ejecutar SO huéspedes en un sistema anfitrión (host)
  + Utilizan en el medio un hypervisor o monitor de máquinas virtuales
  + El SO huésped debe estar soportado en la arquitectura anfitriona
  + Es más eficiente que la emulación (Intel-VT y AMD-V)
* Paravirtualización:
  + Permite correr SOs modificados exclusivamente para actuar en entornos virtualizados
  + Mayor eficiencia que la virtualización

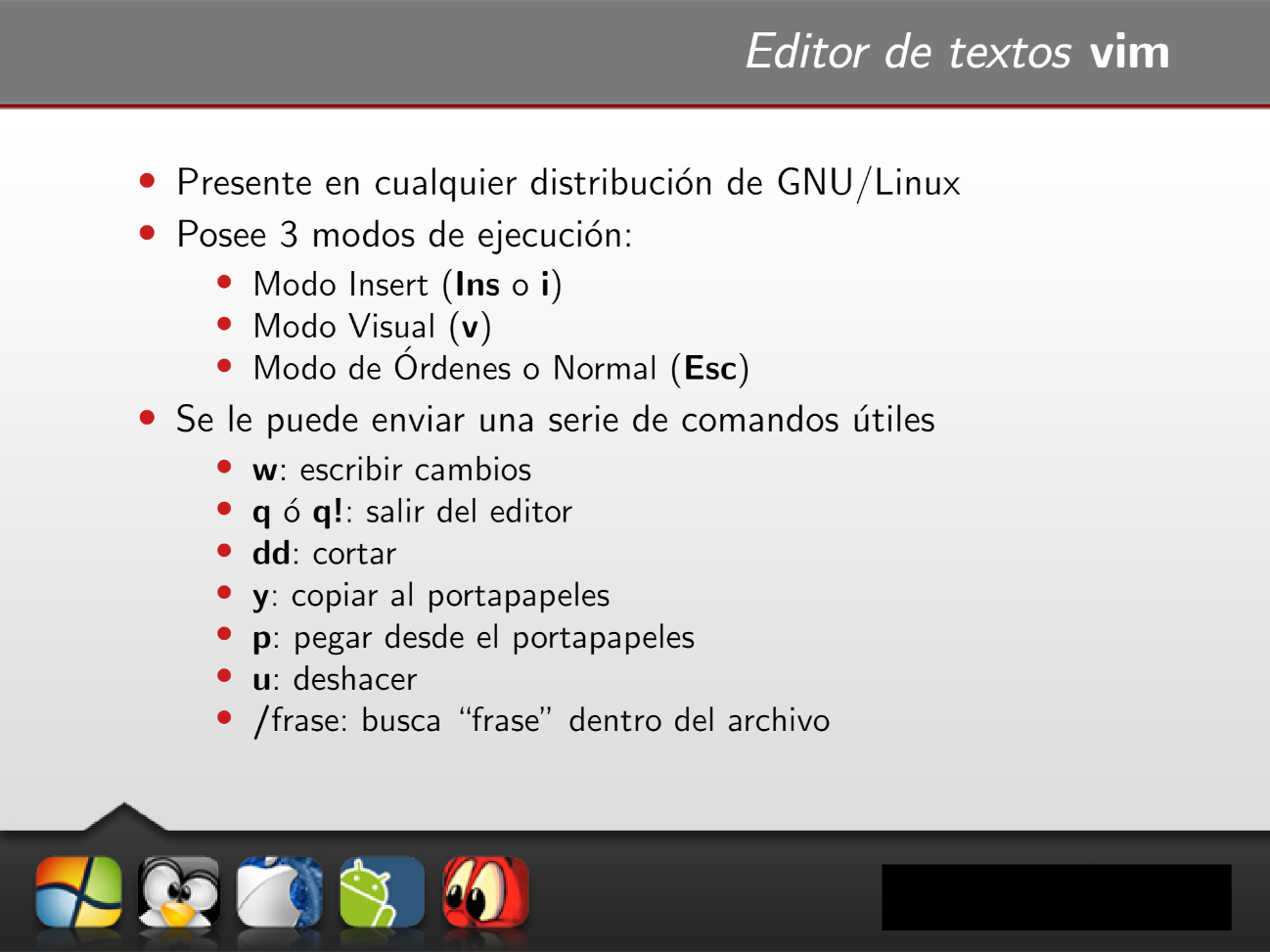
Las principales diferencias entre ellos son:

* Los virtualizadores aprovechan el CPU sobre la que están trabajando, lo cual los hace más veloces.
* En un emulador se puede correr cualquier arquitectura. En un virtualizador solo se puede correr la arquitectura virtualizada

**PRACTICA 2**

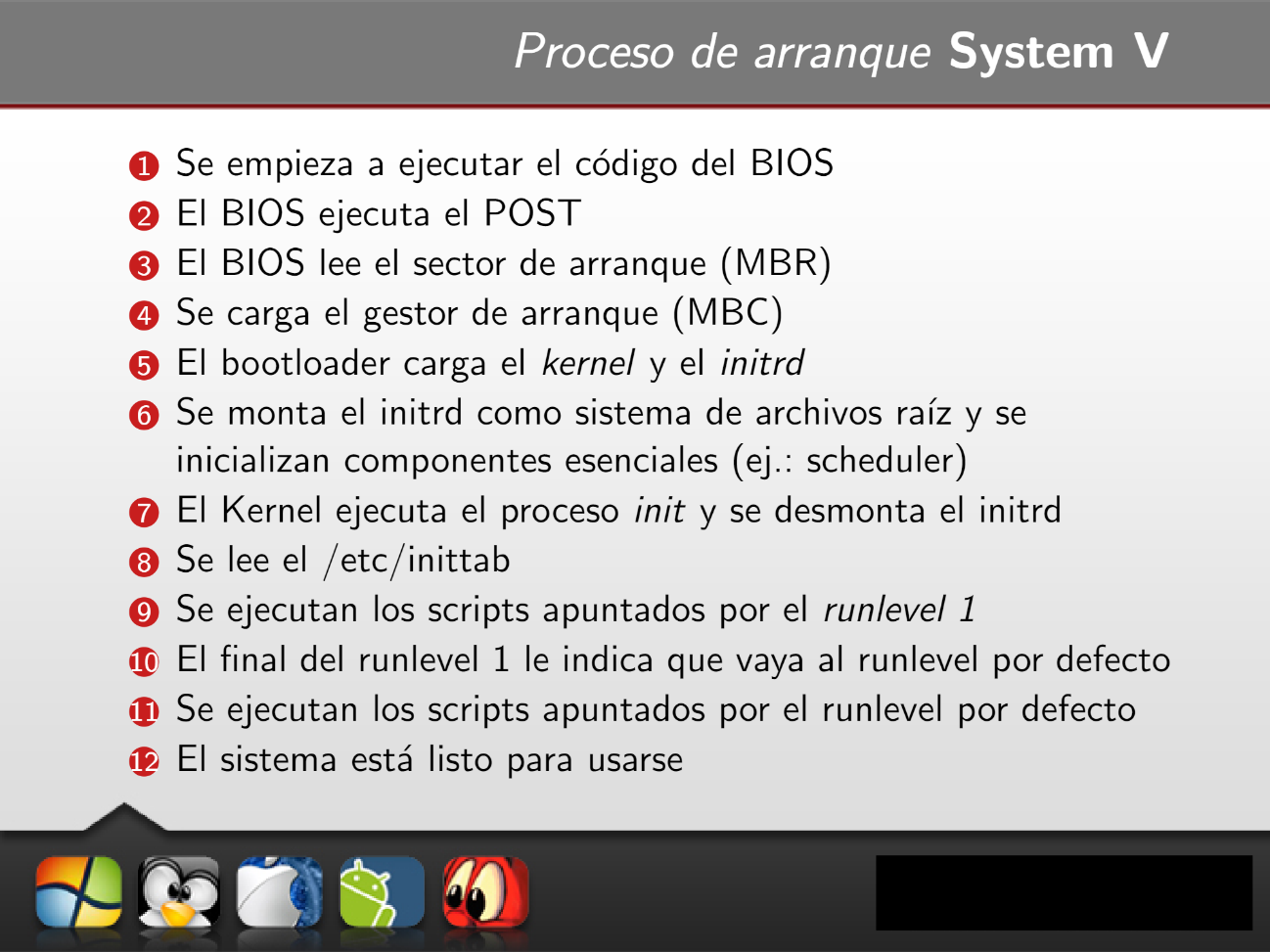
VI





Proceso de Arranque SystemV \*relacionado con pagina 13 (volver)

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.



Proceso INIT

Lo ejecuta el Kernel y su función es cargar todos los subprocesos necesarios para el correcto funcionamiento del SO. Posee PID 1 y se encuentra en /sbin/init. Se lo configura a través del archivo/etc/inittab. No tiene padre y es padre de todos los procesos. Es el encargado demontar los filesystems y de hacer disponibles los demás dispositivos.

pstree: muestra los procesos en ejecución en forma de árbol (padre e hijo) siendo systemd el padre de todos los procesos.

Runlevels

* Es el modo en que arranca linux (3 en redhat, 2 en Debian)
* Un nivel de ejecución es básicamente una configuración de programas y servicios que se ejecutarán orientados a un determinado funcionamiento.
* El proceso de arranque se divide en niveles. Cada uno es responsable de levantar o bajar una serie de servicios.
* Se encuentran definidos en /etc/inittab: Es el archivo de configuración de init. Cuando el sistema se arranca, se verifica si existe un runlevel predeterminado en el archivo/etc/inittab, si no, se debe introducir por medio de la consola del sistema. Después se procede a ejecutar todos los scripts relativos al runlevel especificado.
  + Formato:

*id:niveles\_ejecucion:acción:proceso*

* + - Id: identifica la entrada en inittab (1 a 4 caracteres)
    - Niveles\_ejecucion: el/los nivel de ejecución en los que se realiza la acción
    - Acción: describe la acción a realizar

-wait: se inicia cuando se entra al runlevel e init espera a que termine

- initdefault

- ctrlaltdel: se ejecutará cuando init reciba la señal SIGINT

- off, repawn, once, boot, bootwait, powerwait, otras...

* Proceso: el proceso exacto que será ejecutado

Existen 7 runlevels, y permiten iniciar un conjunto de procesos al arranque o apagado del sistema. Según el estándar:

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

En el caso del modelo runlevel de SystemV, cuando el proceso init arranca, utiliza un fichero de configuración llamado /etc/inittab para decidir el modo de ejecución en el que va a entrar. En este fichero se define el runlevel por defecto (initdefault) en arranque (por instalación en Fedora el 5, en Debian el 2), y una serie de servicios de terminal por activar para atender la entrada del usuario. Después, el sistema, según el runlevel escogido, consulta los ficheros contenidos en /etc/rcn.d, donde n es el número asociado al runlevel (nivel escogido), en el que se encuentra una lista de servicios por activar o parar en caso de que arranquemos en el runlevel, o lo abandonemos. Dentro del directorio encontraremos una serie de script so enlaces a los scripts que controlan el servicio. Cada script posee un nombre relacionado con el servicio, una S o K inicial que indica si es el script para iniciar (S) o matar (K) el servicio, y un número que refleja el orden en que se ejecutarán los servicios.

No todas las distribuciones respetan los estándares.

Comando *init* <x> para cambiar de runlevel

SCRIPTS RC

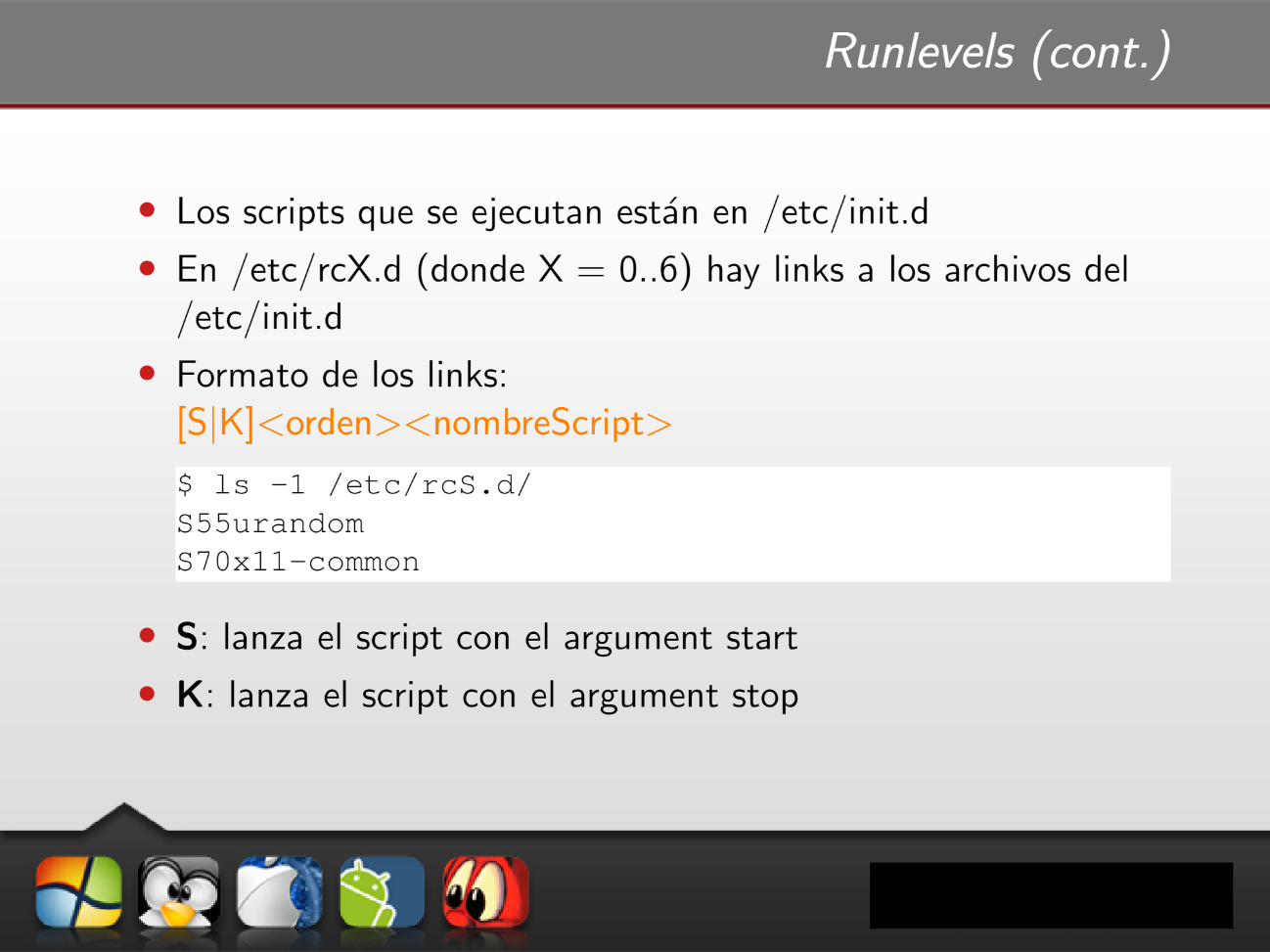
Los scripts RC se encargan de cargar o cerrar los servicios necesarios para que el sistema funcione, de acuerdo con el runlevel que se está iniciando. Estos servicios se encuentran en /etc/init.d/.Sin embargo, no todos los servicios se cargan en todos los runlevels.

Los servicios a cargar se encuentran en el directorio /etc/rcX.d/, donde X es el runlevel a cargar. En realidad, en estos directorios no hay más que enlaces simbólicos a /etc/init.d/ (referencias). Los nombres en estos directorios empiezan con una letra (S o K) seguidos de un número (de 2 dígitos que indican el orden en el que se arrancará el servicio) y el nombre del servicio.

/etc/rc.d/rc cuando entra en un determinado nivel de ejecución realiza las siguientes acciones:

1.Ejecuta, por orden de nombre, todos los scripts que comienzan por K en el directorio correspondiente al nivel, utilizando como argumento para dicho script la opción stop.

2.Ejecuta, por orden de nombre, todos los scripts que comienzan por S en el directorio correspondiente al nivel, utilizando como argumento para dicho script la opción start.



Insserv

El programa insserv se usa para administrar el orden de los enlaces simbólicos que están en «/etc/rcX.d/», resolviendo las dependencias de forma automática. Mejora la performance del arranque en sistemas multiprocesadores.

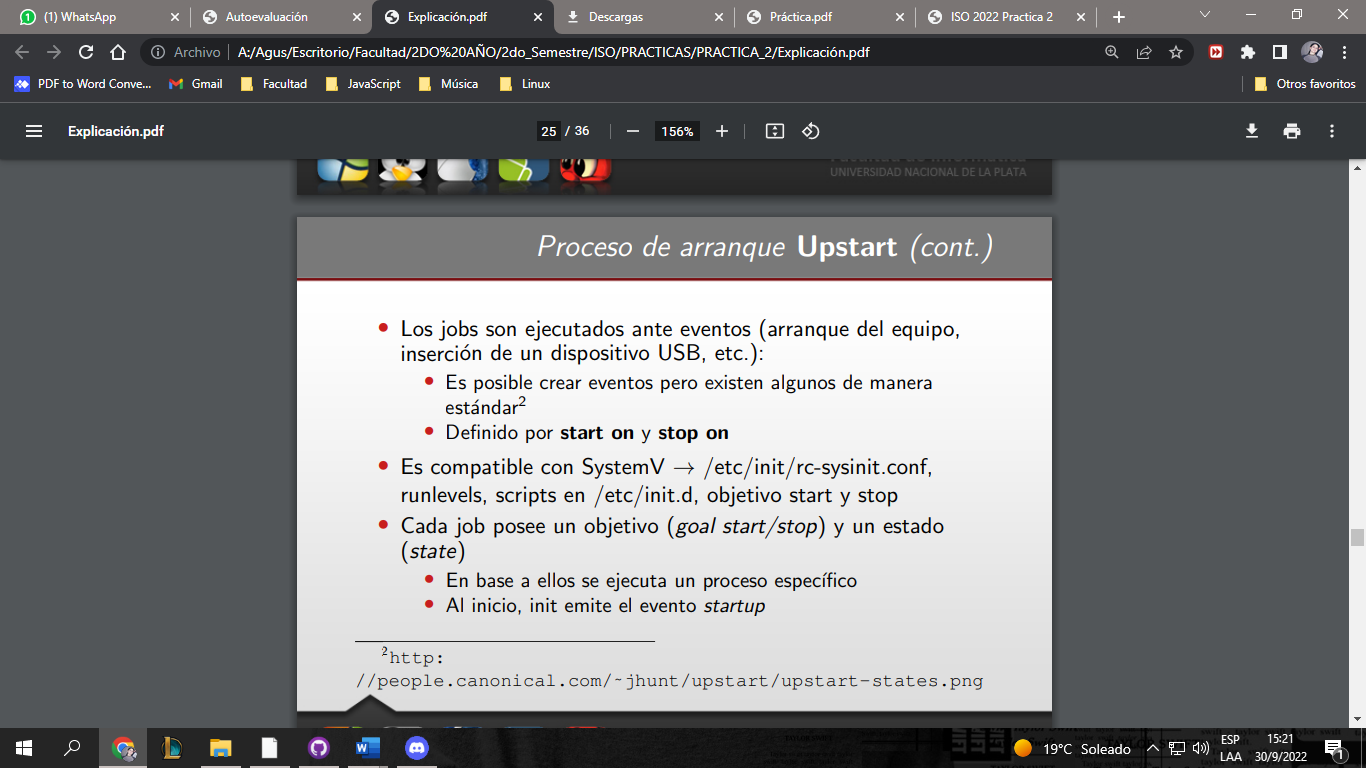
Lo que hace es analizar automáticamente los datos introducidos en la cabecera del script init y guarda los enlaces para los scripts de arranque y parada en los directorios de niveles de ejecución respectivos. Esto permite a cada mantenedor de paquetes especificar en su script de init.d la relación con otros scripts y poder detectar y evitar bucles de dependencias entre scripts así como asegurarse de que todos los scripts se inician en el orden pretendido.

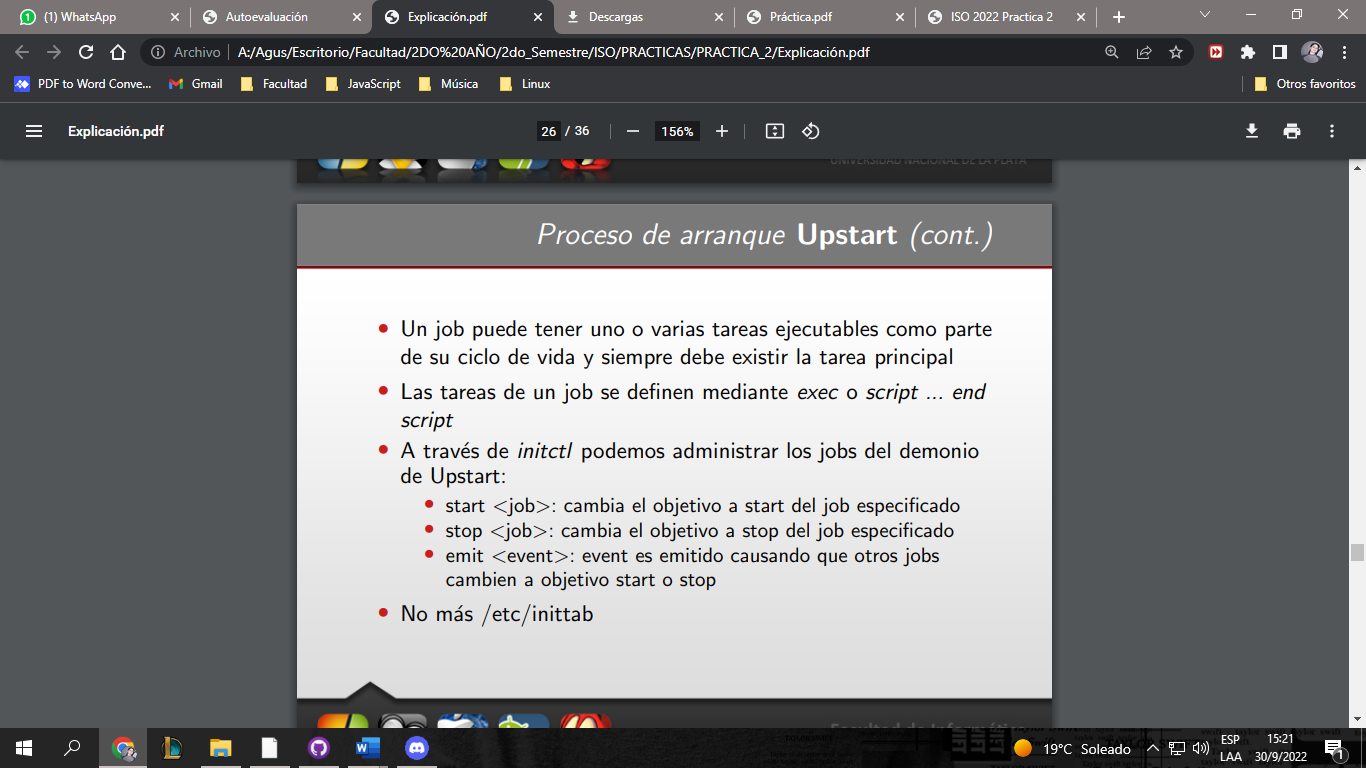
Upstart

* reemplazo basado en eventos, y no en niveles.
* Los servicios se pueden levantar o desactivaren respuesta a ciertos eventos, y este procedimiento permite por ejemplo manejar el reinicio de servicios que mueren de forma inesperada.
* Opera asíncronamente: las tareas y servicios son ejecutados ante eventos (arranque del equipo o inserción de un dispositivo USB) definidos como tareas o Jobs.
* Los jobs se almacenan en el directorio /etc/init.
  + Son scripts en texto plano que definen las acciones a ejecutar. Es compatible con el System V.
  + Dos tipos:
    - Task: ejecución finita (task) → not respawning → exit 0 o uso de stop
    - Service: ejecución indeterminada → respawning

Una de las principales diferencias entre System V y Upstart es que el primero trabaja de forma síncrona mientras que Upstart lo hace de forma asíncrona, es decir, no arranca/para un servicio después de otro sino que puede hacerlo en paralelo. También gestiona las tareas y servicios de inicio cuando el sistema arranca y los detiene cuando el sistema se apaga.

El demonio init tradicional (SystemV) es estrictamente síncrono, bloqueando futuras tareas hasta que la actual se haya completado. Sus tareas deben ser definidas por adelantado, y solo pueden ser ejecutadas cuando el demonio init cambia de estado





REEMPLAZO SCRIPTS RC EN UPSTART

En concreto Upstart ignora el archivo /etc/inittab, en su lugar proporciona un conjunto integrado de scripts de texto plano de arranque que se encuentran en el directorio /etc/init, con nombre ‘servicio’.conf, donde servicio es el programa que init tratará como un job. Los scripts de Upstart ofrecen más acciones que los de SysV, por ejemplo iniciar un servicio siempre que se conecte un determinado dispositivo de hardware.

Systemd

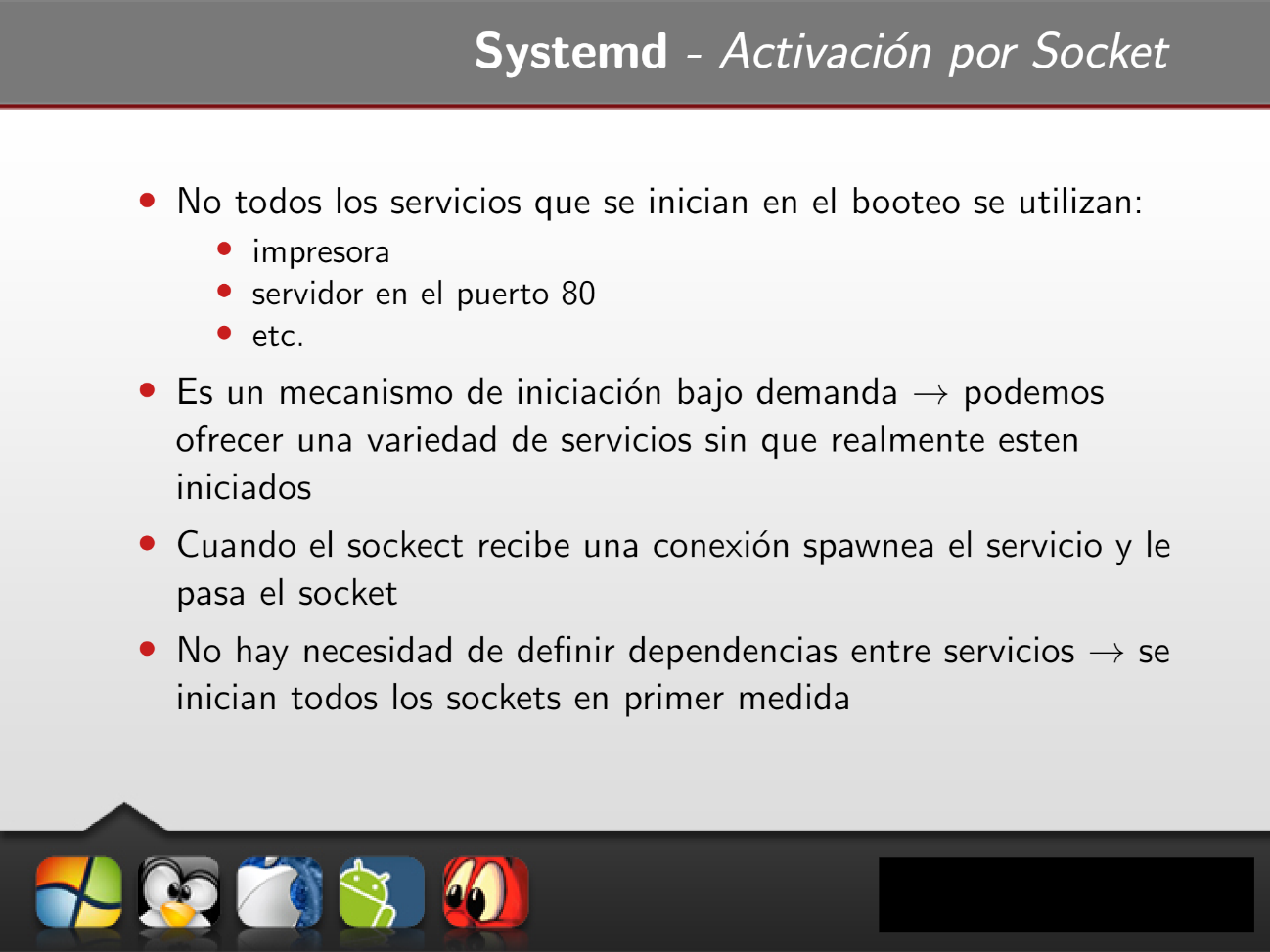
Es un sistema que centraliza la administración de demonios y librerías del sistema

* Mejora el paralelismo de booteo
* Realiza activacion por socket ya que no todos los servicios que se inician en el booteo se utilizan
* Puede ser controlado por systemctl
* Compatible con SysV → si es llamado como init
* El demonio systemd reemplaza al proceso init → este pasa a tener PID 1
* Los runlevels son reemplazados por targets
* Al igual que con Upstart el archivo /etc/inittab no existe más

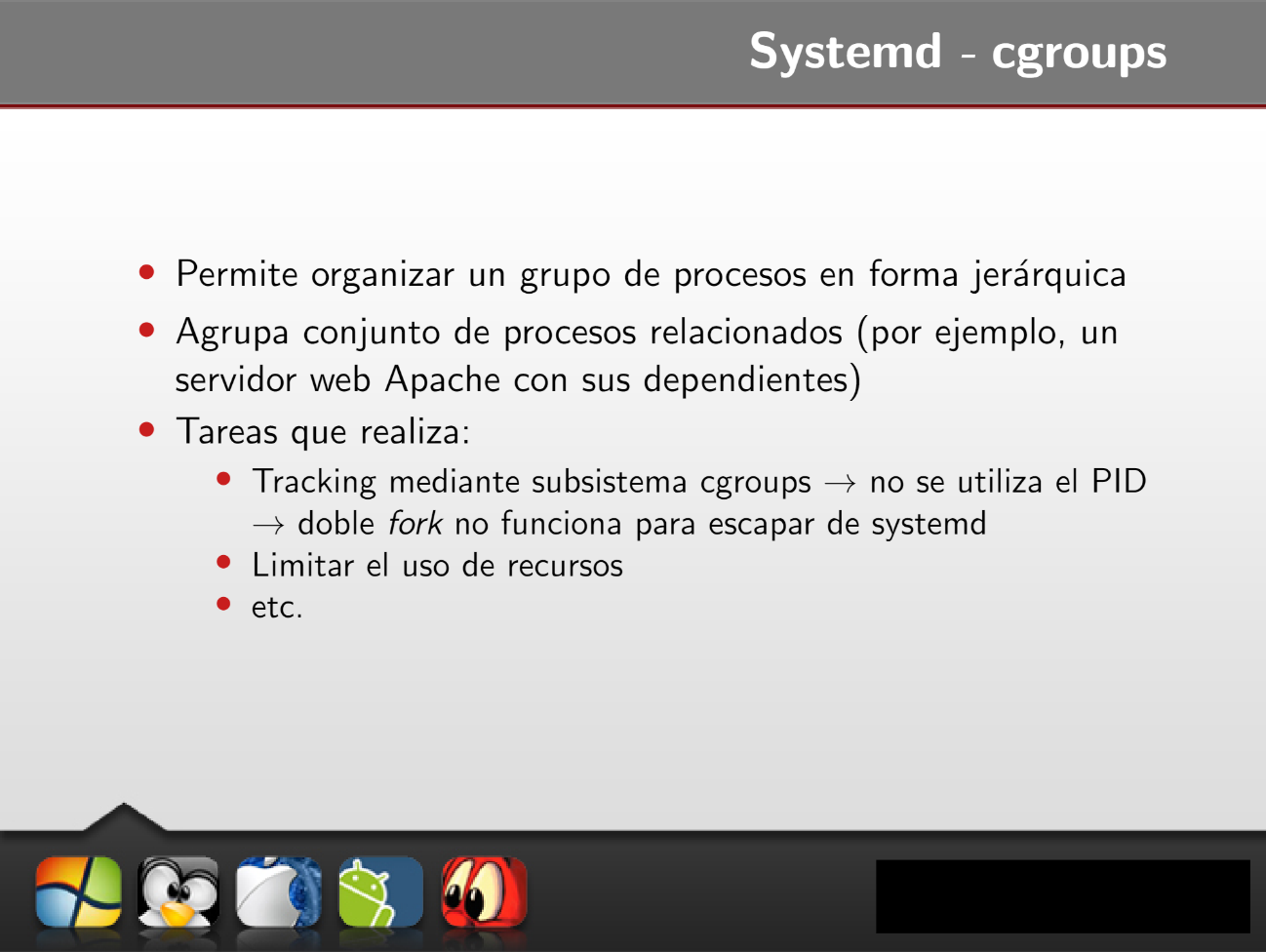
Las unidades de trabajo (dos estados active o inactive) son denominadas units de tipo:

* 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 sincronizaci´on durante el booteo (.target) 🡪 dependencia de unidades
* Snapshot: almacena el estado de un conjunto de unidades que puede ser restablecido más tarde (.snapshot)

**¿A qué hace referencia el concepto de activación de socket en systemd?**

****

**¿A qué hace referencia el concepto de cgroup?**



Es una característica del Kernel de Linux que limita, explica y aísla el uso de recursos (CPU, memoria, E / S de disco, red, etc.) de una colección de procesos.

**Usuarios:**

UID: USER ID

GID: GROUP ID

Si ponemos dos UID iguales a distinto usuario, el S.O. los tomará como una sola cuenta.

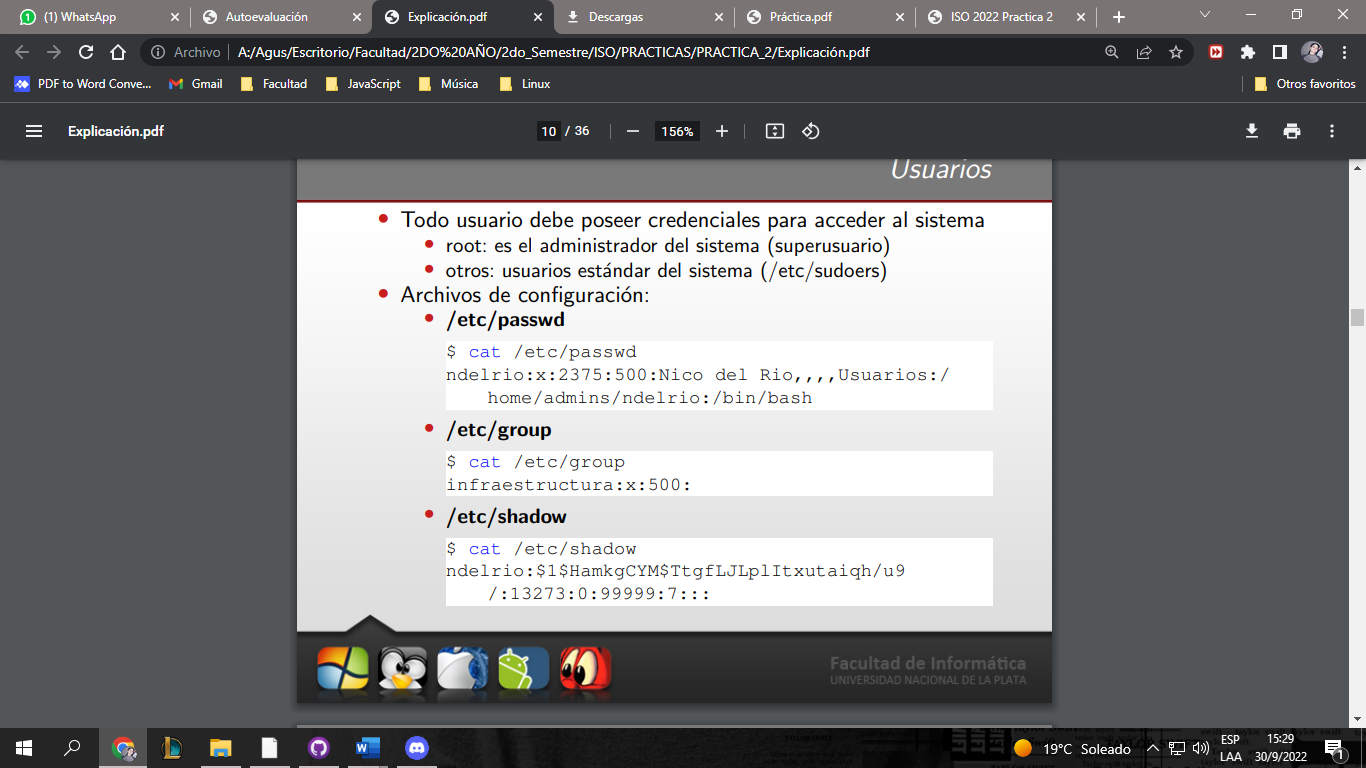
En versiones anteriores podía haber varios usuarios con el mismo UID, luego esto fue cambiado que no de no hacerlo se podía modificar el UID de un usuario a 0 y así este tendrá los mismos permisos que el root

**ROOT** es el administrador. Sólo hay un usuario root, pero pueden haber más de un usuario administrador.

Ser usuario root en un sistema Unix significa tener acceso al directorio raíz, ese donde tenemos instalado todo el sistema operativo. También se le llama ser un Superusuario y en definitiva es tener acceso total al sistema. Algo parecido a tener derechos de administrador en un sistema Windows.

Solo puede haber 1 usuario root pero pueden haber varios admins

UID = 0



* su: permite usar el intérprete de comandos de otro usuario sin necesidad de cerrar la sesión
* groupadd nombre\_grupo : crea un grupo.
* who: dice el usuario activo en ese momento
* useradd <nombreUsuario>:
  + Agrega el usuario
  + Modifica los archivos /etc/passwd
  + Alternativa → adduser
* passwd <nombreUsuario>:
  + Asigna o cambia la contraseña del usuario
  + Modifica el archivo /etc/shadow
* usermod <nombreUsuario>:

-g: modifica grupo de login (Modifica /etc/passwd)

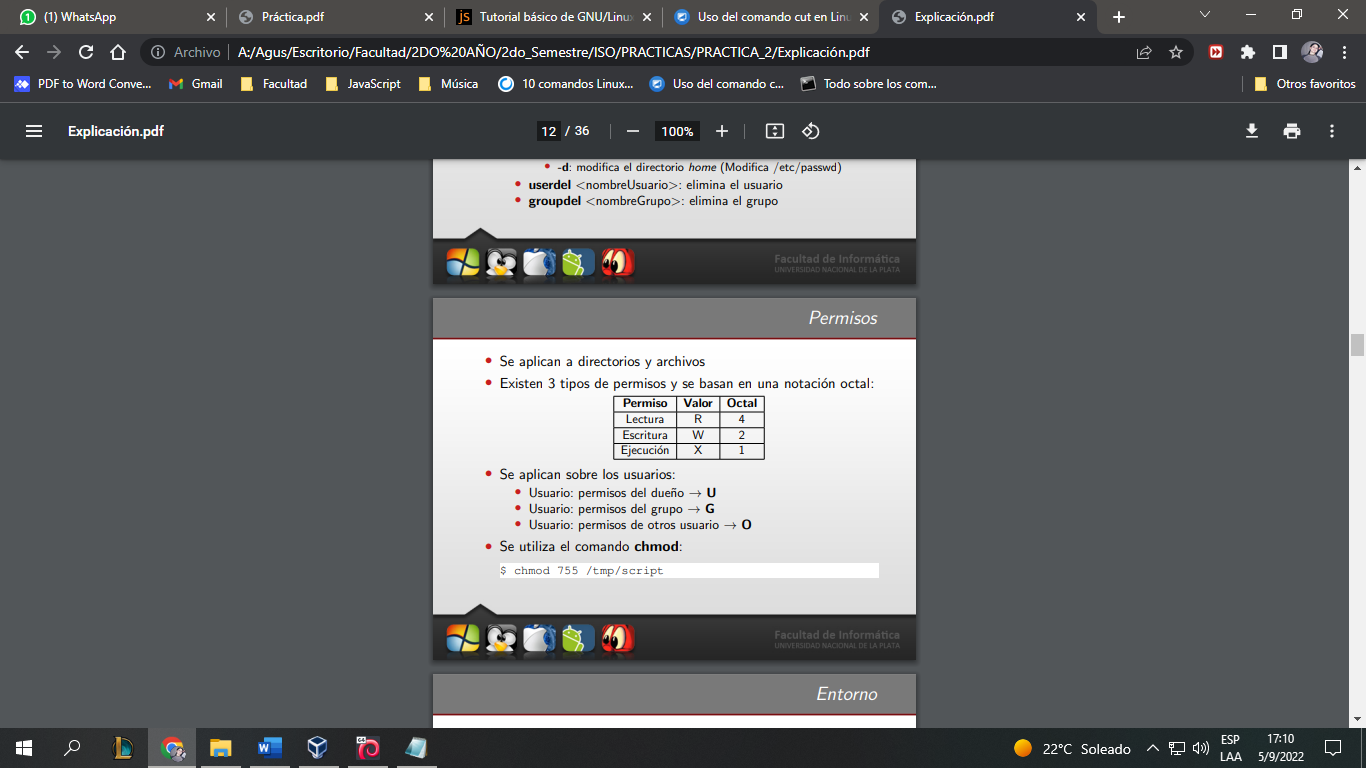
-G: modifica grupos adicionales (Modifica /etc/group)

-d: modifica el directorio home (Modifica /etc/passwd)

* userdel <nombreUsuario>: elimina el usuario
* groupdel <nombreGrupo>: elimina el grupo

PERMISOS

Los permisos están divididos en tres tipos: lectura, escritura y ejecución. Estos permisos pueden ser fijados para tres clases de usuarios: el propietario del archivo o directorio, los integrantes del grupo al que pertenece y todos los demás usuarios.





* r – permiso de lectura (permite abrirlo, copiarlo, etc).
* w – permiso de escritura (permite modificarlo, borrarlo, etc).
* x – permiso de ejecución (si es binario, permite ejecutarlo).

**chmod**: es un comando que permite modificar y especificar quien puede manejar el archivo y cómo puede hacerlo.

chmod +rwx softzone -> añade permiso de lectura, escritura y ejecución al propietario.

chmod g+rwx softzone -> añade permiso de lectura, escritura y ejecución al grupo del usuario propietario.

chmod o+rwx softzone -> añade permiso de lectura, escritura y ejecución al resto de usuarios.

También podemos usar el código numérico para cambiar el nivel de permisos de cualquier archivo de la siguiente manera:

chmod 777 softzone -> permiso rwx para propietario, grupo y resto de usuarios.

chmod 700 softzone -> permiso rwx para propietario, y grupo y usuarios sin permisos.

chmod 327 softzone – permiso de escritura y ejecución para propietario, escritura para grupo y rwx para resto de usuarios.

**chown**: es un comando que permite cambiar el propietario de un archivo o directorio en GNU. Se puede especificar el nombre de usuario o un GID.

chown nombreUsuario archivo1 [archivo2 archivo3…]

chown -R nombreUsuario nombreDirectorio (de manera recursiva la r)

**chgrp**: es un comando que permite cambiar el grupo de usuarios de un archivo o directorio en GNU. Normalmente este tipo de tareas de asignación de permisos se puede realizar con el comando chown pero chgrp maneja una sintaxis más simple para esta tarea

chgrp groupname archivo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Número** | **Binario** | **Lectura (r)** | **Escritura (w)** | **Ejecución (x)** |
| 0 | 000 |  |  |  |
| 1 | 001 |  |  |  |
| 2 | 010 |  |  |  |
| 3 | 011 |  |  |  |
| 4 | 100 |  |  |  |
| 5 | 101 |  |  |  |
| 6 | 110 |  |  |  |
| 7 | 111 |  |  |  |

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

Si, el usuario root, ya que tiene acceso administrativo al sistema (posee permisos de superusuario). Los usuarios normales no tienen este acceso por razones de seguridad.

Los usuarios con acceso a sudo tambien pueden acceder con o sin permisos ya que este comando p permisos de kernel/superusuario

**“full path name”** es la ruta desde el directorio raiz (root) "/" y **“relative path name”** es la ruta iniciando desde la carpeta desde donde "se esta parado"

| Artículo | Descripción |
| --- | --- |
| nombre de ruta absoluto | Rastrea la ruta desde el directorio /(raíz). Los nombres de ruta absolutos siempre comienzan con el símbolo de barra inclinada ( / ). |
| nombre de ruta relativo | Rastrea la ruta desde el directorio actual a través de su padre o sus subdirectorios y archivos. |

Un nombre de ruta absoluto representa el nombre completo de un directorio o archivo desde el directorio /(raíz) hacia abajo. Independientemente de dónde esté trabajando en el sistema de archivos, siempre puede encontrar un directorio o archivo especificando su nombre de ruta absoluto. Ruta absoluta los nombres comienzan con una barra inclinada ( / ), el símbolo que representa el directorio raíz. El nombre de ruta /A/D/9 es el nombre de ruta absoluto para 9. La primera barra inclinada ( / ) representa el directorio /(raíz) , que es el lugar de inicio de la búsqueda El resto del nombre de la ruta dirige la búsqueda a A , luego a D y finalmente a 9 .

Pueden existir dos archivos llamados 9 porque los nombres de ruta absolutos a los archivos dan a cada archivo un nombre único dentro del sistema de archivos.Los nombres de ruta /A/D/9 y /C/E/G/9 especifican dos archivos únicos llamados 9 .

A diferencia de los nombres de rutas completas, los nombres de rutas relativas especifican un directorio o archivo basado en el directorio de trabajo actual. Para los nombres de rutas relativas, puede usar la notación punto punto (..) para moverse hacia arriba en la jerarquía del sistema de archivos. El punto punto (. .) representa el directorio principal. Debido a que los nombres de rutas relativas especifican una ruta que comienza en el directorio actual, no comienzan con una barra inclinada (/). Los nombres de rutas relativas se usan para especificar el nombre de un archivo en el directorio actual o la ruta nombre de un archivo o directorio por encima o por debajo del nivel del directorio actual en el sistema de archivos. Si D es el directorio actual, el nombre de la ruta relativa para acceder a 10 es F/10 . Sin embargo, el nombre de la ruta absoluta siempre es /A/ D/F/10Además, el nombre de la ruta relativa para acceder a 3 es ../../B/3 .

También puede representar el nombre del directorio actual usando la notación punto ( . ) La notación punto ( . ) se usa comúnmente cuando se ejecutan programas que leen el nombre del directorio actual.

**¿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.**

Con pwd.

$HOME

| **Comando** | **Función** |
| --- | --- |
| cd | vuelve a su directorio de login |
| cd ~ | vuelve también a su directorio de login |
| cd / | le lleva al directorio raíz del sistema completo |
| cd /root | le lleva al directorio principal del root, o superusuario, cuenta creada en la instalacion; debe ser el usuario root para accesar este directorio. |
| cd /home | lo lleva a su directorio principal, donde los directorios login de usuario son almacenados |
| cd .. | le traslada a un directorio superior |
| cd ~*otheruser* | le lleva al directorio login del usuario *otheruser*, si *otheruser* le ha dado permiso |
| cd /dir1/subdirfoo | sin tener en cuenta en que directorio esta, este recorrido absoluto le llevara directamente a subdirfoo, un subdirectorio de dir1 |
| cd ../../dir3/dir2 | este recorrido relativo lo llevara dos directorios mas arriba, luego a dir3, luego al directorio dir2. |

* cd: cambia de directorio
* mkdir: crea un directorio
* rmdir: elimina un directorio
* mount: monta un dispositivo
* umount: desmonta un dispositivo
* du: muestra lo que ocupa y el tamaño total de los directorios dentro del directorio donde me encuentro
* df: se usa para chequear el espacio en el disco. Mostrará el almacenamiento
* disponible y utilizado de los sistemas de archivos en tu máquina.
* ln: crea enlaces a archivos y crea un fichero que apunta a otro
* ls: lista los archivos y directorios dentro del entorno de trabajo
* pwd: se utiliza para imprimir el nombre del directorio actual.
* cp: El comando cp se emplea para hacer copias de archivos y directorios en nuestro sistema operativo.
* mv: es usado para mover o renombrar archivos o directorios del sistema de archivos.

PROCESOS

Es un programa en ejecución. Un proceso es una entidad "viva" /se modifica/es dinámico, a diferencia de los programas que son estáticos

PID (Process ID): Es un identificador numérico único para cada proceso.

PPID (Process Parent ID): El identificador del proceso padre.

Todos los procesos tienen estos atributos, además de estos (que son los más importantes, pero no todos) usuario (UID), grupo (GID), prioridad.

con ps -ejH puedo ver el PPID

* Mirar todos los procesos que se están ejecutando en el sistema: ps -ef
* Mirar todos los procesos que está ejecutando un usuario: ps -fu usuario
* Filtrar algunas de las columnas de ps: ps -eopid,tt,user,fname,tmout,f,wchan

# ps -flU usuario1

F S UID PID PPID C PRI NI ADDR SZ WCHAN STIME TTY TIME CMD

4 S usuario1 5083 5082 0 80 0 - 22179 wait 11:45 pts/1 00:00:00 -bash

0 R usuario1 5118 5083 0 80 0 - 22134 - 11:45 pts/1 00:00:00 ps -f

#

El significado de cada una de las columnas es:

UID → Propietario del proceso.

PID → ID del proceso.

PPID → ID del proceso padre.

C → Cantidad de recursos de CPU que el proceso ha utilizado recientemente para que el kernel (núcleo del sistema operativo) establezca la prioridad apropiada a cada proceso.

PRI → Prioridad del proceso.

NI → Valor nice. Un valor positivo indica menor tiempo de CPU.

STIME → Hora de comienzo del proceso.

TTY → Terminal asociado al proceso.

TIME → Tiempo de CPU asociado al proceso.

CMD → Comando ejecutado

Para ver los procesos en sistemas Linux, contamos con el comando ‘ ps ’, que listará (de múltiples formas según las opciones que le pasemos) todos los procesos que se encuentran corriendo en nuestro equipo.

ps aux (muestra todos los procesos del sistema)

ps axjf (que mostrará un árbol jerárquico con la ruta del programa al que pertenece el proceso)

Aquí un ejemplo de filtrado sobre ps para obtener únicamente los procesos pertenecientes a bash: ps aux | grep bash

El estado background se refiere a que, al ser activado desde una terminal, el proceso se ejecuta de manera independiente a la terminal sin "amarrarla" durante el tiempo de proceso.

El estado foreground en cambio, "amarra" la terminal al proceso dejándola sin capacidad de correr otra tarea en esa terminal mientras el proceso se ejecuta.

Cuando un proceso está en ejecución sin que sea mostrado en la terminal se dice que se está ejecutando en el **background** (segundo plano). Si se muestra la ejecución del comando dentro de la terminal se dice que está en el **foreground** (primer plano).

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

Con CTRL + Z es enviar el proceso a segundo plano y queda detenido. Para saber los procesos que están en segundo plano se utiliza el comando jobs.

Para traerlo de background primero se utiliza jobs para saber el número de proceso, y posteriormente fg % seguido del número de proceso (ID).

Los comandos **fg** y **bg** permiten traer un proceso a primer plano o enviarlo a segundo plano respectivamente. Su uso es tan sencillo, como ejecutar la instrucción seguido de % y el número del proceso que obtenemos de jobs.

* traer el proceso a primer plano utilizando fg %2
* devolver el proceso a segundo plano con el atajo de teclado Ctrl+Z
* poner el proceso en marcha una vez que está en segundo plano con bg %2.

Pipe ( | )

El “|” nos permite comunicar dos procesos por medio de un pipe desde la shell

El pipe conecta stdout (salida estandar) del primer comando con la stdin (entrada estandar) del segundo.

* Por ejemplo:
  + $ ls | more
    - Se ejecuta el comando ls y la salida del mismo, es enviada como entrada del comanda more

Se pueden anidar tantos pipes como se deseen

*-Grep busca una palabra o patrón en un lugar, y devuelve todas las coincidencias.*

Redirecciones

En GNU/Linux, al final todo es tratado como si fuera un fichero y como tal, tenemos descriptores de fichero para aquellos puntos donde queramos acceder. Hay unos descriptores de fichero por defecto:

* Entrada estándar (normalmente el teclado).
* Salida estándar (normalmente la consola).
* Salida de error.

Tipo redirección

* Al utilizar redirecciones mediante > (destructiva):
  + Si el archivo de destino no existe, se lo crea.
  + Si el archivo existe, se lo trunca y se escribe el nuevo contenido
* Al utilizar redirecciones mediante >> (no destructiva):
  + Si el archivo de destino no existe, se lo crea
  + Si el archivo existe, se agrega la información al final

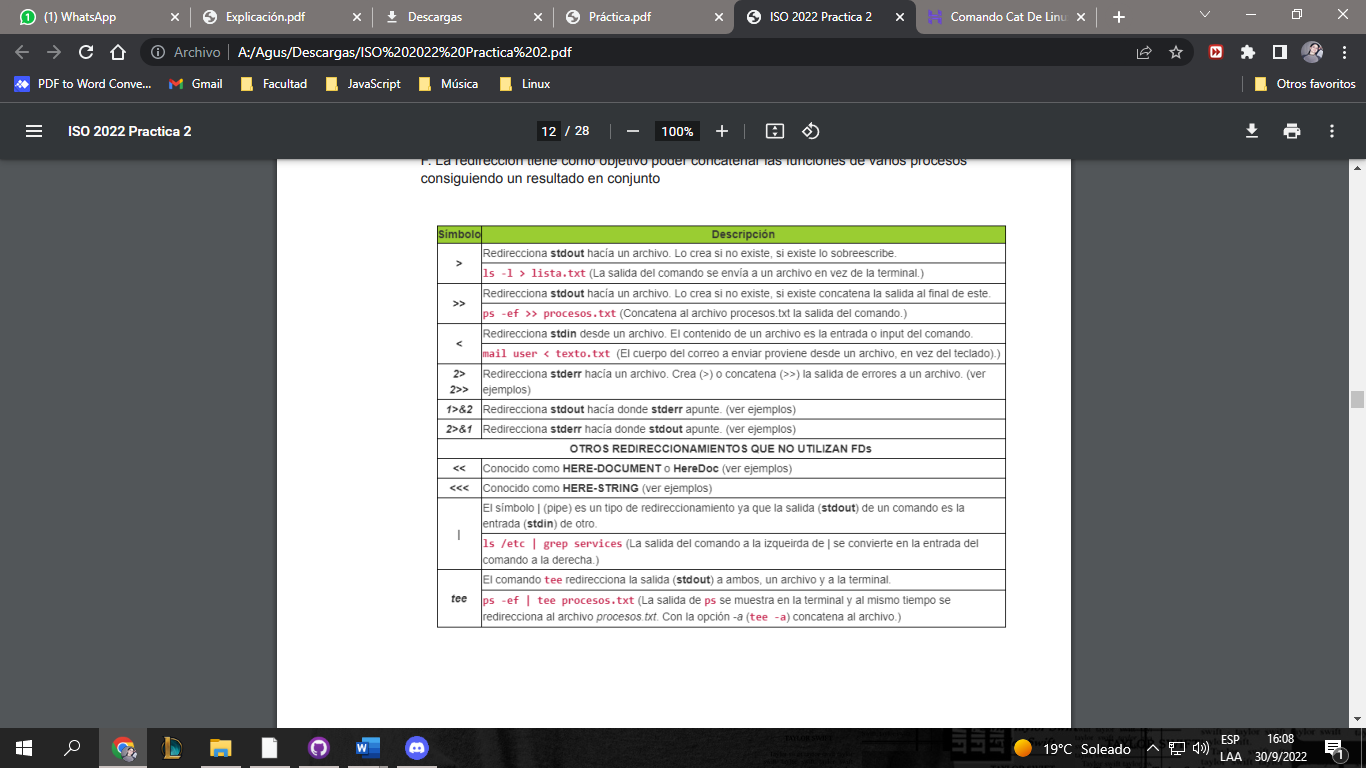
Usando el comando cat puedes crear un archivo rápidamente y agregarle texto. Para hacer eso, usa el operador “>” para redirigir el texto en el archivo.

cat > filename.txt

Si hay algún error, lo guarda en fichero (podría salir un error si no tuviéramos permiso de lectura en el directorio).

En el caso del operador < se redirige la entrada estándar, es decir, el contenido del fichero que especificáramos, se pararía como parámetro al comando.

Si quisiéramos redirigir todas las salidas a la vez hacia un mismo fichero, podríamos utilizar >&. Además, con el carácter & podemos redirigir salidas de un tipo hacia otras, por ejemplo, si quisiéramos redirigir la salida de error hacia la salida estándar podríamos indicarlo con: 2>&1. Es importante tener en cuenta que el orden de las redirecciones es significativo: se ejecutarán de izquierda a derecha.



**kill** es un comando unix/linux que permite enviar señales a uno o varios procesos del sistema a través de la shell (bash, ksh, etc). Las señales más utilizadas suelen ser la de matar un proceso (9 ó SIGKILL), pararlo (TERM) o reiniciarlo (1 ó HUP) pero hay muchas más que pueden ser útiles en ocasiones. El listado completo de señales disponibles puede visualizarse ejecutando kill -l

ps: Muestra información de los procesos activos.

ps [OPCIONES]

Por defecto muestra:

* **PID** : Id del Proceso.
* **TTY** : Terminal.
* **TIME** : Tiempo de ejecución.
* **CMD** : Comando.

**Opciones**

* **a** : Muestra todos los procesos asociados a un TTY.
* **-e** / **-A** : Muestra todos los procesos.
* **x** : Muestra los no asociados.
* -**f**: Muestra el formato largo:
  + **UID** : Usuario que lo ejecutó.
  + **PPID** : Id del proceso padre.
  + **C** : Uso del procesador.
  + **STIME** : Inicio de ejecución.
* **u** : Orientado al usuario:
  + **USER**
  + **% CPU** : uso de procesador.
  + **% MEM** : uso de memoria.
  + **VSZ** : Memoria virtual.
  + **RSS** : Memoria física.
  + **STAT** : Estado.
  + **START** : Iniciado.

Significados de los estados STAT :

* **S**: Esperando (Sleep).
* **R** : Ejecutando (Running)
* **D**: Esperando entrada/salida.
* **T** : Pausa.
* **Z** : No responde (Zombie)

Información adicional de los STAT:

* **s** : Proceso padre
* **l**: Proceso con hilos.
* **+** : En primer plano.

pstree: muestra en forma de árbol los procesos en ejecución.

Con el parámetro “-a”, nos muestra la línea de comandos utilizada. Por ejemplo, si el comando utiliza la ruta a un fichero de configuración.

Por defecto se inhabilita la visualización en el árbol de los nombres repetidos. Para evitar esto, podemos utilizar el parámetro “-c”

Si nos interesa podemos ver el árbol de un proceso específico, de la siguiente manera:

pstree -H [PID]

top: muestra en tiempo real información del sistema. (Por defecto cada 3 segundos)

$ top [-d SEGUNDOS] [-u USUARIO]

htop: con htop podremos obtener un resultado más amigable, no esta instalado por defecto se ha de instalar mediante $ apt-get install htop ... una vez instalado ...

$ htop

killall: mata todos los procesos con un nombre concreto o de un usuario o ambos.

$ killall [-u USUARIO] [-SENAL] [NOMBRE\_PROCESO]

Ejemplos de uso:

Matar todos los procesos con el nombre de "sleep"

$ killall sleep

Matar todos los procesos del usuario "Pepito"

$ killall -u Pepito

Matar todos los procesos del usuario "Pepito" que se llamen "sleep"

$ killall -u Pepito sleep

nice: ejecuta un comando con una prioridad distintas ala de por defecto. Solo los usuarios root pueden establecer prioridades urgentes (negativos).

$ nice -n NUMERO\_PRIORIDAD COMANDO

renice: cambia la prioridad de un proceso ejecutándose. No se puede aumentar la urgencia.

$ renice -n NUMERO\_PRIORIDAD COMANDO

$ nice -10 named (esto bajaría la prioridad de named en -10 unidades (si estaba en -10, pasara a -20) MENOS GENTIL = MAS PRIORIDAD

**¿A qué hace referencia el concepto de empaquetar archivos en GNU/Linux?**

* Empaquetar: Es agrupar en un solo fichero varios ficheros y/o directorios.
* Comprimir: Es reducir el tamaño de un fichero a través del uso de un algoritmo de compresión.

Empaquetar

En linux contamos con el comando tar, que nos permite realizar el proceso de empaquetación, su sintaxis es la siguiente:

tar [opciones] [nombre\_fichero\_tar] [directorio\_origen]

Las opciones más utilizadas son (la versión con un guion es la corta y con dos guiones la larga, pero hacen lo mismo):

-c --create: Crea un nuevo archivo.

-x --extract: Extrae ficheros de un archivo.

-v --verbose: Lista detalladamente los ficheros procesados.

-f \$&fichero\$&: Empaqueta o desempaqueta en o hacia un fichero.

-t --list: Lista los contenidos de un archivo.

Algunos ejemplos de uso son los siguientes:

* Crear un archivo tar llamado edteam.tar con los archivos del directorio cursos.

tar -cf edteam.tar cursos

* Crear un archivo tar llamado edteam.tar con los archivos del directorio cursos mostrando el detalle de los ficheros procesados.

tar -cvf edteam.tar cursos

* Ver el contenido del archivo edteam.tar

tar -tf edteam.tar

Comprimir

Los comandos gzip y gunzip permiten comprimir y descomprimir ficheros respectivamente, su sintaxis básica es:

gzip [archivo\_a\_comprimir]

gunzip [archivo\_a\_descomprimir]

gzip edteam.tar

gunzip edteam.tar.gz

Podemos realizar el proceso de empaquetación y compresión a través de una sola instrucción, agregando la opción -z al comando tar.

Veamos algunos ejemplos:

* Crear un archivo empaquetado y comprimido llamado edteam.tar.gz con los archivos del directorio cursos mostrando el detalle de los ficheros procesados.

tar -cvzf edteam.tar.gz cursos

* Desempaqueta y descomprime el archivo edteam.tar.gz mostrando el detalle de los ficheros procesados.

tar -xvzf edteam.tar.gz

Comprimir con zip

Además de gzip y unzip, podemos comprimir y descomprimir a través de zip y unzip respectivamente, este formato de compresión es el más utilizado en sistemas operativos Windows.

Su sintaxis básica es la siguiente:

zip -r [nombre\_fichero\_zip] [directorio\_a\_comprimir]

unzip [nombre\_fichero\_zip]

* Crear un archivo comprimido llamado edteam.zip con los archivos del directorio cursos:

zip -r edteam.zip cursos

* Descomprimir el archivo edteam.zip

unzip edteam.zip

|  |  |
| --- | --- |
| **Empaquetar archivos:** | tar cvf *nombre.tar directorio\_o\_nombres\_archivos* |
| **Desempaquetar archivos:** | tar xvf *nombre.tar* |
| **Comprimir archivos:** | gzip *nombre\_archivo\_a\_comprimir* |
| **Descomprimir archivos:** | gzip -x *nombre\_archivo\_a\_descomprimir* |

**¿Pueden comprimirse un conjunto de archivos utilizando un único comando?**

No, se deben comprimir de a uno

tar : permite empaquetar varios archivos en uno solo, sin comprimirlos. Tambien sirva para desempaquetar.

Ej: tar –vcf nombre\_archivo.tar nombre\_carpeta\_a\_empaquetar

Ej: tar –vxf mi\_archivo.tar

grep : muestra líneas que concuerdan con un patrón.

gzip : comprime o expande ficheros.

zgrep : busca una expresión regular en ficheros posiblemente comprimidos.

wc : sirve para contar líneas, palabras y caracteres que contiene un archivo.

FIND

find / -(i)name “nombre” 🡪 un archivo en particular

ej find / -name my-file 🡪 busca archivo llamado my-file

ej find / -iname my-file 🡪 busca archivo llamado my-file sin distinguir mayúsculas de minusculas

find / -not -name my-file 🡪 busca archivo sin my-file en el nombre

find / -name “\*.txt” 🡪 características similares

/ a partir del directorio raíz

. a partir de donde estoy parado

~ directorio personal

Búsqueda por tipo:

d – directorio o carpeta

f – archivo normal

l – enlace simbólico

c – dispositivos de caracteres

b – dispositivos de bloque

find / -type d 🡪 todos los directorios en mi sistema de archivos

find / -type f -name my-file 🡪 esto buscará archivos llamados my-file, excluyendo directorios o enlaces.

find / -name “LEAME”

EJEMPLOS COMANDOS IMPORTANTES

cp ∗ /home −R 🡪 intenta copiar todo lo que esta en el directorio donde se esta parado a home. El -R indica que se haga lo mismo con los archivos y subcarpetas internos

rm ∗ 🡪 borra todos los archivos del directorio en el que se esta parado

Desempaqueta y descomprime en otro directorio.

tar -xvf misLogs.tar **-C** /home/agusnfr/dir1

tar -xzvf misLogs.tar.gz **-C** /home/agusnfr/dir2