




Sistemas Operativos

 Teoría

SOA (1er Módulo)

 Práctica 1

 Práctica 2



Práctica 1

Los sistemas operativos son de uso general.

GNU es una iniciativa con el objetivo de crear un sistema operativo Unix de libre distribución. Luego se crea la FSF (free software foundation) para financiar el proyecto GNU. Desde esta iniciativa nacen varios programas de software, como por ejemplo GCC y otras librerías características de Unix.

Para implementar el Kernel, se fusiona el proyecto de Linus Torvalds, que venía trabajando en un kernel llamado Linux. De ahí nace GNU/Linux.

GNU se basa en 4 libertades principales: libertad de usar el programa, estudiarlo, distribuirlo y mejorarlo.

Las características del software libre son:

- Cuando se obtiene puede ser usado, copiado, estudiado, modificado y redistribuido libremente.
- Generalmente son de costo nulo, pero el software libre no es software gratuito.
- Generalmente se distribuye junto con su código fuente.
- Las fallas se corrigen de forma rápida y están a cargo de la comunidad.
- Libertad de usar el programa, estudiarlo, distribuirlo y mejorarlo.

Las del software propietario son inversas a las del libre. Por ejemplo, tienen un costo asociado, no se lo puede distribuir, las fallas están a cargo del propietario, etc.

Características Generales

- El sistema operativo GNU/Linux es multiusuario, esto quiere decir que permite manejar múltiples usuarios y permisos.
- Es multitarea y multiprocesador.
- Altamente portable
- Variados intérpretes de comandos, algunos programables.
- Filesystem: todo es un archivo (incluyendo dispositivos y directorios).

Componentes

Sus componentes más importantes son:

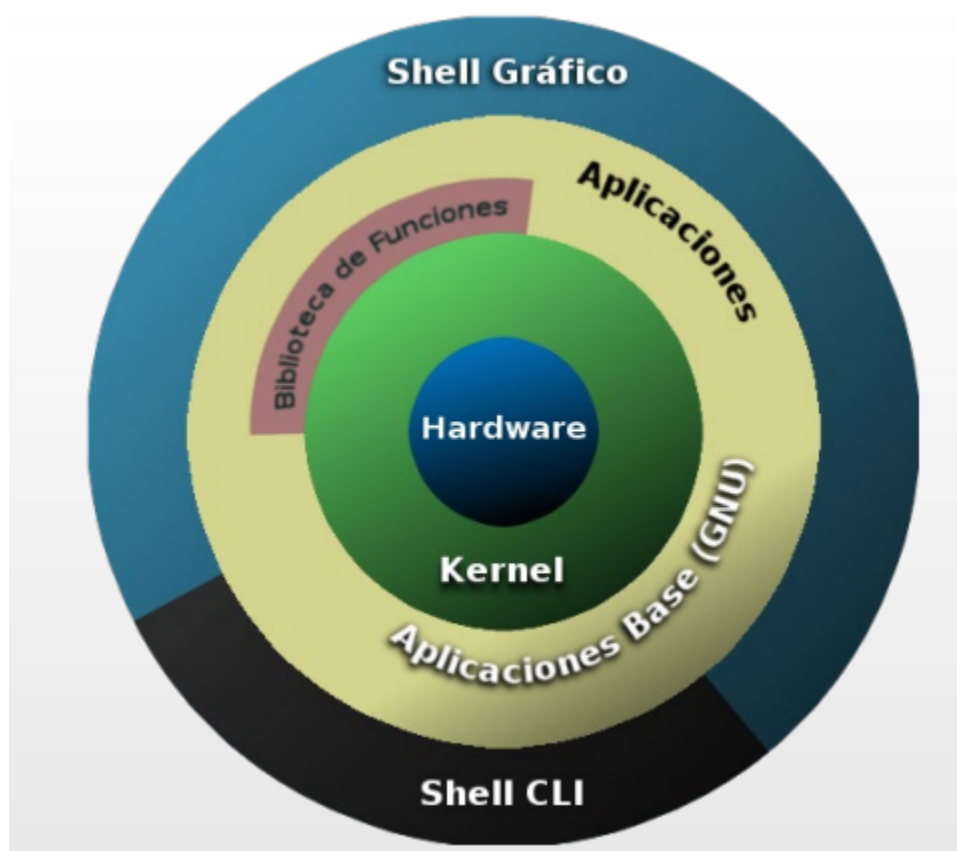
1. **Kernel de Linux:** El kernel es el corazón del sistema operativo y actúa como intermediario entre el hardware y el software. Administra los recursos del sistema, como la memoria, la CPU, los dispositivos de entrada y salida, y permite que los programas se comuniquen con el hardware de la computadora de manera eficiente.
2. **Shell y terminal:** El shell es una interfaz de línea de comandos que permite a los usuarios comunicarse con el sistema operativo escribiendo comandos. La terminal es la aplicación que proporciona un entorno en el que se ejecuta el shell. Bash es uno de los shells más comunes en sistemas GNU/Linux.
3. **Sistema de archivos:** El sistema de archivos organiza y almacena todos los datos en la computadora, incluidos los archivos del sistema, los documentos y

las aplicaciones. En un sistema GNU/Linux, todo es un archivo. GNU/Linux utiliza una jerarquía de sistema de archivos estándar (FSH) que define la estructura de los directorios y su propósito.

Todos estos componentes trabajando en conjunto hacen posible que el entorno de GNU/Linux sea eficiente y cómodo de utilizar e interactuar.

Estructura

La estructura de un sistema GNU/Linux se compone de esta forma:



Kernel

El kernel de GNU/Linux es el núcleo del sistema operativo que gestiona los recursos de hardware y proporciona una interfaz entre el hardware y el software. Es parte esencial de la distribución GNU/Linux y es responsable de controlar los procesos, la memoria, los dispositivos de hardware y la comunicación entre componentes.

Sus funciones principales son:

- Administración de memoria: Asigna y administra la memoria del sistema para que las aplicaciones puedan ejecutarse de manera eficiente.
- Planificación de procesos: Decide qué procesos se ejecutan en qué momento y durante cuánto tiempo, asegurando un uso eficiente de la CPU.
- Gestión de dispositivos: Controla la interacción entre el hardware y el software, permitiendo que los programas se comuniquen con dispositivos como discos duros, impresoras, tarjetas de red, etc.
- Sistema de archivos: Proporciona acceso a los sistemas de archivos, permitiendo que los programas almacenen y recuperen datos en el disco.
- Comunicación interprocesos: Facilita la comunicación y el intercambio de datos entre diferentes procesos en el sistema.
- Seguridad: Implementa medidas de seguridad para controlar el acceso a recursos del sistema y proteger la integridad de los datos.

Una característica a mencionar del Kernel es que se pueden tener más de uno instalado en la misma máquina, permitiendo al usuario elegir entre una u otra en el momento de arranque. El Kernel se localiza en el directorio `/boot` (imagen y archivos relacionados).

El Kernel de GNU/Linux es monolítico, esto significa que todas las funciones esenciales y controladores de dispositivos están presentes en el kernel como un solo bloque de código en memoria. Aunque se pueden cargar y descargar módulos de kernel dinámicamente para agregar funcionalidades específicas, la mayor parte del kernel reside en el espacio del núcleo y se ejecuta en modo privilegiado.

Nomenclatura de 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

Shell

El Shell es una interfaz de línea de comandos que permite a los usuarios interactuar con un sistema operativo ingresando comandos de texto. El shell toma los

comandos ingresados por el usuario, los interpreta y los ejecuta, permitiendo el control del sistema, la administración de archivos, la ejecución de programas y más.

Sus principales funciones son:

- Ejecución de programas: Permite ejecutar programas y utilidades del sistema.
- Administración de archivos: Proporciona comandos para crear, copiar, mover, eliminar y administrar archivos y directorios.
- Automatización: Permite la creación de scripts que ejecutan una serie de comandos de manera secuencial.
- Gestión de procesos: Permite la administración de procesos en ejecución, como iniciar, detener y monitorear programas.
- Manipulación de texto: Ofrece herramientas para manipular y procesar texto, como búsqueda, reemplazo y formateo.
- Personalización del entorno: Permite configurar variables de entorno, alias y otras opciones para personalizar la experiencia del usuario.

Algunos ejemplos de shells son Bash, Zsh y Fish.

Los comandos que utiliza la shell de forma propia se ejecutan dentro del mismo, y no tienen **ubicación** física en el sistema de archivos. Por otro lado, los comandos externos se encuentran en diversos directorios, listados en la variable de entorno PATH.

Es posible definir un intérprete de comandos distinto para cada usuario. Cada usuario puede tener su propio shell predeterminado definido en el archivo `/etc/passwd`. Sin embargo, no todos los usuarios pueden realizar esta tarea, ya que cambiar el shell predeterminado generalmente requiere permisos de administración del sistema.

File System (sistema de archivos)

Un sistema de archivos es una estructura que organiza y almacena datos en un dispositivo de almacenamiento, como discos duros, unidades flash, tarjetas SD, etc. Define:

- **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.

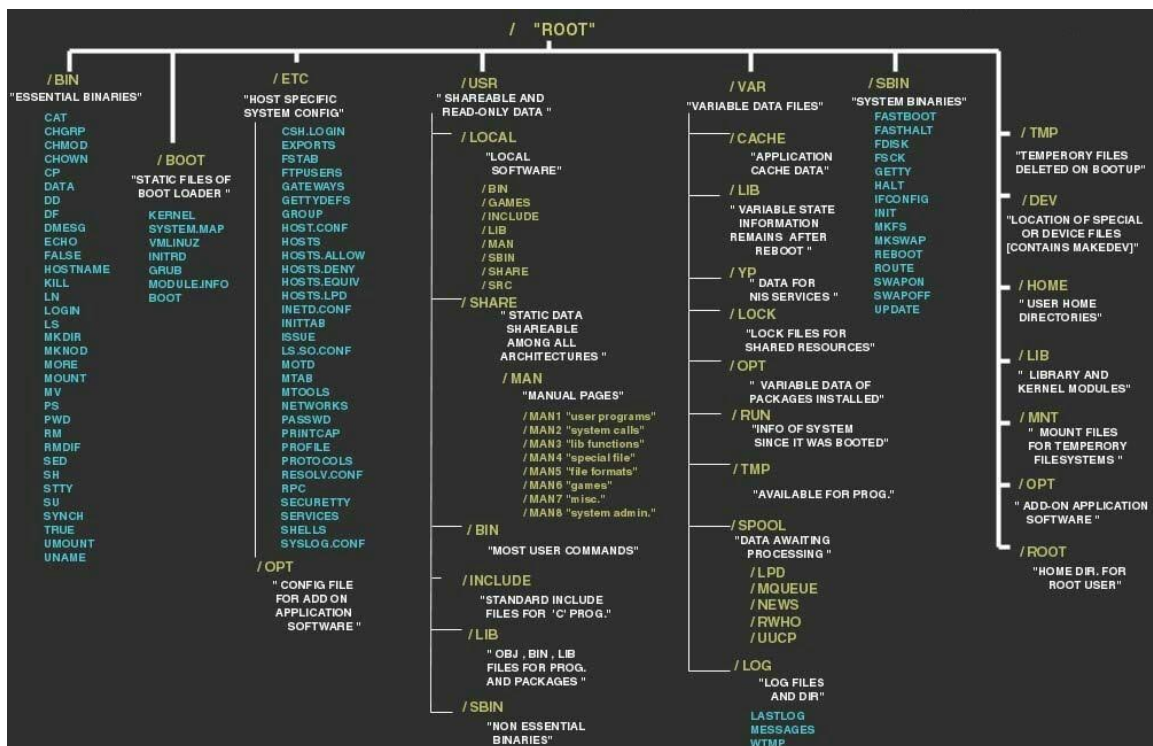
- **Manejo de la memoria secundaria:** Cómo se administra el espacio para los archivos en memoria secundaria.
- **Mecanismos de integridad:** con qué métodos se garantiza la incorruptibilidad del archivo.

El sistema que adopta GNU/Linux es el Extended (v2, v3 y v4).

GNU/Linux es compatible con una variedad de sistemas de archivos, entre los cuales se incluyen Ext4, Btrfs, XFS, F2FS, NTFS, FAT32, exFAT. Es posible visualizar particiones del tipo FAT y NTFS en GNU/Linux. Aunque NTFS es un sistema de archivos de Microsoft, el soporte para leer y escribir particiones NTFS se ha implementado en GNU/Linux.

La estructura de directorios en GNU/Linux sigue el estándar FHS (Filesystem Hierarchy Standard), que define la organización y los propósitos de los directorios principales. Algunos directorios importantes son:

- **/bin** : Contiene archivos ejecutables esenciales que se necesitan para el arranque y el funcionamiento del sistema, como los comandos básicos del shell.
- **/boot** : Aquí se encuentran los cargadores de inicio y los archivos de inicio del sistema.
- **/dev** : donde se encuentran montados todos los dispositivos físicos. **/dev/null** es un archivo especial que elimina todo lo que se escribe en él.
- **/etc** : Almacena archivos de configuración del sistema y del software instalado.
- **/home** : Contiene los directorios personales de los usuarios, donde se almacenan sus archivos y configuraciones.
- **/lib** : librerías de los paquetes instalados.
- **/var** : Contiene archivos variables, como registros de sistema, archivos de caché y otros datos que cambian durante la operación del sistema.
- **/usr** : Almacena los archivos y programas de usuario, incluyendo binarios, bibliotecas, archivos de encabezado y documentación.
- **/tmp** : Es un directorio temporal donde los programas y usuarios pueden crear y almacenar archivos temporales.
- **/root** : Es el directorio personal del usuario root (superusuario).
- **/opt** : Se utiliza para instalar software opcional y adicional.



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.

Distribuciones

Una distribución de GNU/Linux es una versión específica del sistema operativo GNU/Linux que incluye el kernel de Linux y una variedad de software, herramientas y utilidades para crear un sistema completo y funcional. Cada distribución puede tener un enfoque diferente en términos de objetivos, público objetivo, programas y paquetes preinstalados y características. Ejemplos de distribuciones son Ubuntu, Debian, Fedora y Kali, entre otras.

Las principales diferencias entre las distribuciones son:

- **Objetivos y público:** Algunas distribuciones se centran en ser amigables para usuarios principiantes, mientras que otras son más adecuadas para usuarios avanzados o administradores de sistemas.
- **Ciclo de lanzamiento:** Algunas distribuciones tienen ciclos de lanzamiento rápidos con nuevas características frecuentes, mientras que otras optan por lanzamientos más estables y menos frecuentes.

- **Selección de software:** Las distribuciones pueden incluir diferentes conjuntos de software predeterminados, entornos de escritorio y herramientas.
- **Filosofía y licencias:** Algunas distribuciones pueden priorizar el software completamente libre y de código abierto, mientras que otras pueden permitir ciertos componentes propietarios.
- **Sistema de gestión de paquetes:** Cada distribución puede utilizar un sistema de gestión de paquetes diferente, lo que afecta cómo se instala, actualiza y administra el software.
- **Comunidad y soporte:** Las distribuciones varían en términos de tamaño de comunidad, documentación y soporte disponible.

Particiones

Una partición es una división lógica de un disco físico que se utiliza para organizar y administrar los datos. Cada partición actúa como si fuera un disco independiente y puede contener un sistema de archivos y datos. Los sistemas operativos y los datos se almacenan en estas particiones.

Los tipos principales de particiones son:

- **Partición primaria:** Es la partición principal en un disco y puede contener un sistema de archivos y un sistema operativo.
- **Partición extendida:** Una partición que se utiliza para contener particiones lógicas adicionales. Solo puede haber una partición extendida por disco.
- **Partición lógica:** Se encuentra dentro de una partición extendida y se utiliza para crear unidades lógicas separadas dentro de una unidad física.

Ventajas: el particionamiento permite separar y organizar diferentes tipos de datos, permite la instalación de varios sistemas operativos en la misma máquina, facilita la realización de copias de seguridad y restauraciones y puede mejorar la eficiencia y el rendimiento al optimizar el uso del disco.

Desventajas: el particionamiento puede llevar a una fragmentación del espacio en disco, puede ser complicado administrar múltiples particiones y el tamaño incorrecto de una partición puede resultar en desperdicio o falta de espacio.

Las particiones se pueden manejar mediante aplicaciones de software. Hay dos tipos:

- **Destruyivas:** crean y eliminan particiones (fdisk).

- **No Destructivas:** crean, eliminan y modifican particiones (fips, gparted). Las distribuciones permiten realizar este tipo de modificaciones en la etapa de instalación.

Identificación

En GNU/Linux, las particiones se identifican mediante nombres de dispositivos. Para discos IDE, los nombres suelen comenzar con `/dev/hd` seguido de una letra (por ejemplo, `/dev/hda` , `/dev/hdb`). Para discos SCSI y SATA, los nombres suelen comenzar con `/dev/sd` seguido de una letra (por ejemplo, `/dev/sda` , `/dev/sdb`).

Para instalar GNU/Linux se necesitan como mínimo dos particiones:

1. **Partición de arranque (boot):** Es una partición primaria que almacena los archivos de arranque del sistema operativo. Puede contener el cargador de arranque y otros archivos necesarios para el inicio.
2. **Partición de raíz (/):** Esta partición contiene el sistema de archivos raíz y todo el sistema operativo.

Bootstrap

El proceso de bootstrap consiste en el **arranque** del sistema operativo. Consta de varios pasos:

1. La computadora se enciende y el BIOS o UEFI realiza pruebas de hardware (POST) y busca dispositivos de arranque.
2. El BIOS o UEFI encuentra el gestor de arranque en el MBR o en la partición de arranque y lo carga en la memoria.
3. El gestor de arranque presenta opciones al usuario si hay varios sistemas operativos instalados.
4. El gestor de arranque carga el kernel del sistema operativo seleccionado en la memoria y le pasa el control.
5. El kernel inicia el sistema operativo, monta sistemas de archivos y carga los controladores de dispositivos.

BIOS (Basic I/O System)

El BIOS es un firmware (software integrado en el hardware) de bajo nivel que se encuentra en la placa base de una computadora. Su tarea principal es buscar y

ejecutar el MBC. También proporciona instrucciones básicas para la comunicación entre el sistema operativo y el hardware.

UEFI (Unified Extensible Firmware Interface)

UEFI es un reemplazo moderno del BIOS. A diferencia del BIOS, UEFI es más flexible y puede funcionar en sistemas de 32 y 64 bits. Proporciona una interfaz más rica y segura para el firmware, y puede ser utilizado para funciones como la administración de arranque y la seguridad.

El UEFI fue una especie de alianza entre varias compañías con el objetivo de modernizar el proceso de arranque. Con modernizar nos referimos a que es más veloz, más comprensible y más segura (se implementa el Secure Boot).

MBR y MBC

El MBR (Master Boot Record) es una estructura ubicada en el primer sector del disco duro (cilindro 0, cabeza 0, sector 1) que contiene información crítica para el proceso de arranque. A su vez, el MBR contiene un programa llamado MBC. El MBC (master boot code) posee el código necesario para iniciar el sistema operativo. Seguido del MBC el MBR contiene la tabla de particiones, cuya función es saber dónde y cuánto espacio ocupa cada partición. El MBC examina esta tabla, se fija cual es la partición booteable (donde está el kernel) y la ejecuta (lleva el Kernel a memoria).

GPT (GUID Partition Table)

GPT es un formato de tabla de particiones moderno que reemplaza al antiguo MBR. GPT permite particiones más grandes y más de cuatro particiones primarias. Proporciona mejor protección contra la corrupción de la tabla de particiones.

Gestor de Arranque

Un gestor de arranque es un programa que permite seleccionar y cargar el sistema operativo al encender la computadora. Estos gestores pueden instalarse en el MBR o en una partición específica y permiten la selección de sistemas operativos o configuraciones de arranque.

Shutdown (parada)

Es el proceso que ocurre cuando apagamos o reiniciamos una computadora. Ocurre de la siguiente manera:

1. Se detienen los servicios y procesos en orden.
2. Se desmontan los sistemas de archivos.
3. Se detiene el kernel y se apagan los dispositivos.
4. La computadora se apaga o reinicia.

POSIX

POSIX (Portable Operating System Interface) es un conjunto de estándares y especificaciones desarrollado para garantizar la portabilidad y la compatibilidad entre sistemas operativos Unix y sistemas similares. El objetivo principal de POSIX es definir una interfaz estándar que permita que las aplicaciones escritas para un sistema operativo Unix se ejecuten en otros sistemas Unix sin necesidad de modificaciones significativas.

La idea detrás de POSIX es permitir que las aplicaciones escritas para un sistema operativo Unix puedan ser fácilmente portadas a otros sistemas Unix, ya que cumplir con las especificaciones POSIX asegura que las interfaces y funcionalidades esenciales sean consistentes entre diferentes sistemas.

Bootstrap (proceso de arranque)

Discos

Los discos tienen espacios reservados.

Uno de estos espacios se llama MBR (espacio privilegiado), que significa Master Boot Record. A su vez, el MBR contiene un programa llamado MBC. El MBC (master boot code) posee el código necesario para iniciar el sistema operativo. Seguido del MBC el MBR contiene la tabla de particiones, cuya función es saber dónde y cuánto espacio ocupa cada partición.

El MBC examina la tabla de particiones, se fija cual es la partición booteable (donde está el kernel) y la ejecuta (lleva el Kernel a memoria).

Cuando prendemos la máquina, se ejecuta el BIOS, que luego ejecuta el MBC.

Los tipos de particiones del MBR son:

- Primarias: se restringen a 4 debido al espacio acotado.
- Extendida: una de las 4 primarias puede ser la extendida, que sirve para contener en su interior unidades lógicas. Solo puede existir una por disco. Tiene su propia lista de particiones.

- Lógica: se contienen dentro de las extendidas, ocupan una fracción de la partición extendida, o su totalidad. Este tipo de particiones se conectan como una lista enlazada.



Práctica 2

Arranque con System V

El proceso de arranque en los sistemas GNU/Linux utiliza el esquema System V, que consta de varios pasos:

1. Comienza a ejecutarse el código de la **BIOS**.
2. El BIOS ejecuta el **POST** (Power-On Self-Test, serie de pruebas para verificar que el HW funcione).
3. El BIOS lee el sector de arranque **MBR**.
4. Se carga el gestor de arranque **MBC**.
5. El bootloader carga el **kernel** y el **initrd**.
6. Se monta el **initrd** como sistema de archivos raíz, y se inicializan componentes esenciales.
7. El kernel ejecuta el proceso **init** y se desmonta ~~initrd~~.
8. Se lee el `/etc/inittab` (define los runlevels).
9. Se ejecutan los scripts del **runlevel 1**.
10. Se va al **default runlevel**.
11. Se ejecutan los **scripts** del default runlevel.
12. El sistema se puede utilizar.

Proceso INIT (paso 7)

El proceso INIT (inicialización) es el proceso principal que se ejecuta como el primer proceso después de que el kernel haya cargado. Su objetivo es inicializar y gestionar el sistema operativo y los servicios esenciales. INIT es el ancestro de todos los demás procesos en el sistema y se encarga de la secuencia de inicio,

carga de controladores de dispositivos y configuración del entorno necesario para que otros procesos y servicios se ejecuten correctamente.

Se encuentra en `/sbin/init`, y se lo configura mediante `/etc/inittab`.

Runlevels

Definen diferentes estados operativos del sistema. Cada nivel de ejecución representa un conjunto específico de servicios y configuraciones que determinan el comportamiento del sistema en un momento dado. Se encuentran definidos en `/etc/inittab`. El objetivo principal de los runlevels es controlar qué servicios se inician o detienen en diferentes situaciones, como el arranque del sistema, el apagado, el reinicio o la transición entre modos de funcionamiento.

En el estándar SystemV, hay 7 runlevels (del 0 al 6), cada uno para un uso específico:

- **Runlevel 0:** Apagado del sistema. El sistema se detiene completamente.
- **Runlevel 1:** Modo de usuario único o "single-user mode". Se utiliza para realizar tareas de mantenimiento o recuperación del sistema con un mínimo de servicios.
- **Runlevel 2:** Modo multiusuario sin soporte de red.
- **Runlevel 3:** Modo multiusuario con soporte de red. Generalmente, este es el nivel de ejecución predeterminado en sistemas Linux basados en SystemV init.
- **Runlevel 4:** No está definido en el estándar SystemV init y, por lo general, se deja para la personalización local.
- **Runlevel 5:** Modo gráfico o multiusuario con interfaz gráfica de usuario (GUI). En muchas distribuciones, este nivel de ejecución inicia el entorno de escritorio.
- **Runlevel 6:** Reinicio del sistema. Similar a un apagado, pero reinicia el sistema en lugar de apagarlo por completo.

La configuración de los runlevels y sus correspondientes servicios se define en los directorios de inicio específicos de cada nivel, que suelen estar ubicados en `/etc/rc.d` o `/etc/init.d` en muchas distribuciones. Los enlaces simbólicos a scripts de inicio y parada se crean en estos directorios para habilitar o deshabilitar servicios en cada nivel.

No todas las distribuciones de Linux respetan estrictamente este estándar SystemV init. Algunas distribuciones han adoptado sistemas de inicio alternativos como Systemd, Upstart o otros, que ofrecen un enfoque más flexible y moderno para la

gestión de servicios y no necesariamente siguen la misma estructura de runlevels. Esto significa que la forma en que se gestionan los servicios y se inicia el sistema puede variar según la distribución.

Archivo `/etc/inittab`

El archivo `/etc/inittab` ejerce un papel importante en los sistemas Linux que utilizan SystemV init para el proceso de inicio. Su finalidad principal es configurar la inicialización del sistema, definir los runlevels y asignar acciones específicas a eventos del sistema.

La estructura de información en el archivo `/etc/inittab` solía estar compuesta por líneas que seguían un formato específico. Cada línea constaba de cuatro campos principales:

1. **ID de proceso (Process ID):** Un identificador único para la entrada que generalmente consta de una letra y un número (por ejemplo, "c1", "1:2345", etc.).
2. **Nivel de ejecución (Runlevel):** El runlevel al que se aplicaba la entrada. Esto especificaba cuándo y en qué nivel de ejecución se debía ejecutar la acción o comando.
3. **Tipo de evento (Event Type):** El tipo de evento que activaría la acción, como "initdefault" para el nivel de ejecución predeterminado, "respawn" para reiniciar un proceso si se detiene, "once" para ejecutar una acción una sola vez, entre otros.
4. **Comando o acción:** El comando o acción que se ejecutaría cuando se cumplieran las condiciones especificadas en los campos anteriores.

Si queremos cambiar de runlevel, podemos hacerlo mediante el comando `sudo init X`, donde X es el runlevel al que queremos cambiar. Este cambio es solamente temporal, y cuando el sistema se reinicie volverá al runlevel que el archivo `/etc/inittab` tenga marcado como default.

Scripts RC

Los scripts RC son scripts de inicio y parada que se utilizan en sistemas Linux, especialmente en aquellos que siguen el esquema de SystemV init, para iniciar y detener servicios y configuraciones específicas durante el proceso de arranque o apagado del sistema. Su finalidad principal es gestionar servicios y configuraciones de manera ordenada y controlada.

Se almacenan en los directorios `/etc/init.d/` y `/etc/rc.d/` (tiene enlaces simbólicos a `init.d`), y se organizan en función del nivel de ejecución al que se destinan. Por ejemplo, los scripts de inicio para el nivel de ejecución 3 se almacenan en un directorio específico para ese nivel.

Cuando el sistema GNU/Linux arranca o se detiene, el sistema determina qué scripts RC ejecutar mediante el archivo `/etc/inittab`. Este archivo especifica qué scripts se ejecutan en cada runlevel.

El orden para llamar los scripts RC está determinado por las prioridades numéricas (Sxx o Kxx) en el nombre de los scripts. Los scripts que comienzan con "S" se ejecutan durante el proceso de inicio, y los que comienzan con "K" se ejecutan durante el proceso de apagado. El número "xx" en el nombre del script indica la secuencia en la que se ejecutarán, siendo los números más bajos ejecutados primero.

Por ejemplo, en el nivel de ejecución 3, un script llamado `S10servicio` se ejecutaría antes que un script llamado `S20otroservicio`. Esto permite un control preciso de la inicialización y detención de servicios y configuraciones en el sistema.

Insserv

Es una herramienta que se utiliza para gestionar la secuencia de inicio de los scripts RC y las dependencias entre ellos. Su objetivo principal es automatizar y simplificar la configuración de los runlevels y las relaciones de dependencia entre los scripts.

Las ventajas de `insserv` respecto a un arranque tradicional manual son:

1. **Gestión de dependencias:** `insserv` puede identificar y gestionar automáticamente las dependencias entre los scripts RC, lo que simplifica la configuración del inicio y evita problemas relacionados con el orden incorrecto de ejecución.
2. **Consistencia y organización:** Ayuda a mantener una estructura de directorios y nombres de scripts coherente y organizada, lo que facilita la administración del sistema.
3. **Automatización:** Permite automatizar la configuración del inicio y evita la necesidad de realizar cambios manuales en archivos de configuración, lo que reduce el riesgo de errores humanos.

Upstart

Upstart es un sistema de inicio que surge como evolución de SystemV init. Tenía como objetivo mejorar el proceso de arranque del sistema y la gestión de servicios introduciendo la gestión de servicios de forma paralela.

El proceso de arranque del sistema en Upstart se basa en eventos y tareas. En lugar de depender estrictamente de niveles de ejecución (runlevels), Upstart se centra en responder a eventos y desencadenar tareas en función de esos eventos. Algunas de las características clave de Upstart incluyen:

- **Eventos:** Los eventos son disparadores que ocurren en el sistema, como el inicio del sistema, la detección de hardware, la conexión de una red, etc.
- **Tareas:** Las tareas son las acciones que se ejecutan en respuesta a eventos. Estas tareas pueden ser scripts de inicio o detención de servicios, o cualquier otra acción que deba llevarse a cabo cuando ocurra un evento específico.
- **Dependencias:** Upstart permite definir dependencias claras entre servicios y tareas, lo que facilita la gestión de las secuencias de inicio y garantiza que las dependencias se resuelvan adecuadamente.

Jobs

En Upstart, los scripts RC (Run Control) de SystemV, que se utilizaban para iniciar y detener servicios y configuraciones en el proceso de arranque del sistema, son reemplazados por archivos de configuración llamados "job files" o archivos de trabajo. Estos archivos se utilizan para definir cómo debe gestionarse un servicio o tarea en Upstart. Los job files se encuentran generalmente en el directorio

`/etc/init/`.

Los archivos de trabajo (job files) de Upstart tienen una sintaxis específica y describen las acciones que Upstart debe realizar en respuesta a eventos específicos, como el inicio del sistema o la detección de una red. En lugar de utilizar scripts shell personalizados como en SystemV init, Upstart utiliza estos archivos de trabajo para definir las tareas y las dependencias del servicio.

Upstart VS SystemV init

1. **Modelo de inicio:** SystemV init se basa en niveles de ejecución (runlevels) y scripts de inicio y parada específicos para cada nivel, mientras que Upstart se basa en eventos y tareas para gestionar el inicio y la detención de servicios.
2. **Gestión de dependencias:** Upstart ofrece una gestión de dependencias más avanzada y flexible que SystemV init. Upstart permite definir relaciones de

dependencia explícitas entre servicios, lo que garantiza un orden de inicio adecuado y una resolución de dependencias más confiable.

3. **Eficiencia:** Upstart se diseñó para ser más eficiente en la gestión de eventos y tareas en comparación con SystemV init. Esto puede llevar a un inicio más rápido del sistema y una mayor capacidad de respuesta en situaciones de alta carga.
4. **Compatibilidad:** SystemV init es ampliamente compatible con los scripts de inicio y parada heredados, lo que facilita la transición desde sistemas más antiguos. Upstart, aunque puede ser compatible con scripts de SystemV, tiende a requerir una configuración específica para aprovechar al máximo sus capacidades.

Systemd

Systemd es un sistema de inicio y administración de servicios que se utiliza en muchas distribuciones de Linux modernas para gestionar el proceso de arranque del sistema y la administración de servicios y tareas en segundo plano.

Systemd se diferencia de sus predecesores en varios aspectos clave, incluyendo:

- Paralelización del inicio de servicios.
- Activación por socket.
- Reemplazo de proceso init por demonio systemd.
- Reemplazo de runlevels por targets.
- `/etc/inittab` obsoleto (igual que en Upstart).

También introduce el concepto de unidades de trabajo (tienen dos estados, activa o inactiva), y pueden ser de distintos tipos:

- **Service:** controla un servicio en particular.
- **Socket:** representan sockets de red o archivos de socket.
- **Target:** establece puntos de sincronización durante el booteo.
- **Snapshot:** almacena el estado de ciertas unidades, que puede ser reestablecido más tarde.

Activación de socket en systemd

Característica que permite a los servicios ser activados y gestionados de manera eficiente a través de sockets (puertos de red o archivos de sistema) cuando sea necesario, en lugar de mantenerlos en ejecución de forma continua. Cuando un servicio es activado por socket, systemd lo inicia solo cuando se recibe una conexión entrante en el socket asociado al servicio. Esto es particularmente útil para servicios que no necesitan ejecutarse todo el tiempo, sino solo cuando hay una solicitud para ellos.

La activación de socket en systemd ayuda a optimizar el uso de recursos del sistema, ya que los servicios solo se inician cuando realmente se necesitan, y se detienen cuando no se están utilizando. Esto puede mejorar el rendimiento y la eficiencia del sistema en entornos con múltiples servicios.

cgroups

Los cgroups permiten organizar grupos de procesos jerárquicamente. Esto es útil ya que nos permite limitar y aislar el uso de recursos para una utilización más eficiente y ordenada.

Usuarios

La información de los usuarios se guarda en varios archivos y directorios. Los archivos principales utilizados para almacenar información de usuarios son:

/etc/passwd : Este archivo contiene información básica de los usuarios, como sus nombres de usuario (**login**), identificadores de usuario (**UID**), identificadores de grupo primario (**GID**), nombres completos y rutas de inicio de sesión.

/etc/shadow : Este archivo almacena las **contraseñas cifradas** de los usuarios, así como información de seguridad adicional relacionada con las contraseñas. El acceso a este archivo suele estar restringido a los usuarios y procesos privilegiados para garantizar la seguridad de las contraseñas.

/etc/group : Contiene información sobre los **grupos** del sistema, incluyendo nombres de grupo, identificadores de grupo (GID) y listas de usuarios que pertenecen a cada grupo.

UID (User Identifier): El UID es un número único que se asigna a cada usuario en un sistema GNU/Linux. Es utilizado internamente por el sistema operativo para identificar y gestionar a los usuarios. Cada usuario tiene un UID único en el sistema,

y los UIDs suelen ser valores enteros positivos. Si hablamos de contextos diferentes, se puede dar que hayan UIDs iguales, pero esto puede llegar a traer problemas.

GID (Group Identifier): El GID es un número único que se asigna a cada grupo en un sistema GNU/Linux. Al igual que con los UIDs, los GIDs son utilizados por el sistema para identificar y gestionar los grupos a los que pertenecen los usuarios. Cada grupo tiene un GID único en el sistema.

Root (superusuario)

El usuario root tiene privilegios máximos en el sistema y puede realizar cualquier acción, incluyendo la modificación de archivos de sistema críticos, instalación de software, gestión de usuarios y grupos, y configuración del sistema en su totalidad. Debido a estos privilegios, el usuario root tiene un control absoluto sobre el sistema y, por lo tanto, debe ser utilizado con precaución.

Este usuario tiene un UID de 0, que está reservado solo para el root. La existencia de un solo usuario root es una medida de seguridad para garantizar que no haya ambigüedad en cuanto a quién tiene el control total del sistema.

Permisos

Existen 3 tipos de permisos, aplicables en cada usuario sobre un determinado archivo:

Permiso	Valor	Octal
Lectura	R	4
Escritura	W	2
Ejecución	X	1

Pueden haber permisos de dueño o grupo.

Para cambiar los permisos, se utiliza el comando **chmod**, junto con la notación octal. Un ejemplo sería

```
$ chmod 755 /tmp/script
```

Donde se habilitan todos los permisos para el dueño con el primer dígito, para el grupo el segundo dígito y para todos los demás el último 5.

Path Names

Existen 2 tipos de path names:

- **Relative:** es una especificación de la ubicación de un archivo o directorio en relación con el directorio actual. No comienza con una barra inclinada ("/") y es relativo al directorio actual desde el cual se está realizando la referencia.
- **Full:** Un full path name (nombre de ruta completa) es una especificación completa de la ubicación de un archivo o directorio que comienza desde la raíz del sistema de archivos. Se inicia con una barra inclinada ("/") y continúa describiendo todos los directorios que deben atravesarse para llegar al archivo o directorio deseado.

Procesos

Es una instancia en ejecución de un programa o una tarea. Cada proceso tiene su propia área de memoria y recursos asignados, lo que le permite ejecutarse de forma independiente de otros procesos en el sistema. Los procesos son fundamentales para el funcionamiento del sistema operativo, ya que permiten la multitarea y la ejecución concurrente de programas.

IDs

- **PID (Process ID):** El PID es un número único que identifica de manera exclusiva a cada proceso en el sistema. Los PIDs son asignados por el kernel del sistema operativo y se utilizan para gestionar y controlar los procesos. No puede haber dos procesos con el mismo PID en un momento dado en el mismo sistema.
- **PPID (Parent Process ID):** El PPID es el PID del proceso padre que creó el proceso actual. Indica la relación jerárquica entre procesos. El proceso padre es aquel que inició o creó el proceso hijo.

No todos los procesos tienen un PPID explícito en sistemas GNU/Linux. Los procesos raíz del sistema (inicializados por el kernel) generalmente tienen un PPID

de 1, que es el PID del proceso "init" o su sucesor (como "systemd" en sistemas más modernos). Los procesos que se inician a partir de la línea de comandos o desde el entorno de usuario generalmente tienen como padre al shell que los inició, lo que implica que tienen un PPID correspondiente.

Atributos

- **Estado del proceso:** Indica si el proceso está en ejecución, suspendido, detenido, etc.
- **Prioridad de CPU:** Un valor que influye en la asignación de tiempo de CPU al proceso.
- **Usuario propietario:** El usuario bajo el cual se ejecuta el proceso.
- **Grupo propietario:** El grupo al que pertenece el proceso.
- **Recursos asignados:** Cantidad de memoria, CPU y otros recursos asignados al proceso.
- **Ruta del ejecutable:** La ubicación del programa o comando que el proceso está ejecutando.

Planos de procesos

Foreground (primer plano): Un proceso se ejecuta en primer plano cuando ocupa la terminal actual y bloquea la entrada del usuario hasta que termine. Al ejecutar un comando en primer plano, la terminal no acepta nuevos comandos hasta que el proceso actual haya finalizado o haya sido suspendido. Se puede ejecutar un proceso en primer plano simplemente escribiendo el comando sin el símbolo `&`.

Background (segundo plano): Un proceso se ejecuta en segundo plano cuando se ejecuta sin ocupar la terminal actual. Esto significa que el proceso se ejecuta en el fondo, y permite seguir utilizando la terminal para ejecutar otros comandos sin esperar a que el proceso en segundo plano termine. Para iniciar un proceso en segundo plano es necesario agregar el símbolo `&` al final del comando, o al suspender un proceso que estaba en primer plano utilizando `Ctrl+Z` y luego enviándolo al segundo plano con el comando `bg`.

Scripting

Variables

```
nombre="pepe" # para crear una variable
echo ${nombre}
echo $nombre # para acceder
```

No pueden empezar con un número.

```
nombre=Carlos
echo "Hola $nombre" # Hola Carlos
echo Hola ${nombre} # Hola Carlos
nombre=5
echo "Hola $nombre" # Hola 5
```

Arreglos

```
arreglo_a=() # arreglo vacio

arreglo_b=(1 2 3 4 5) # arreglo declarado

arreglo_b[2]=15 # posicion concreta

elemento=${arreglo_b[2]} # asignacion

echo ${arreglo_b[@]}
echo ${arreglo_b[*]}
# todos los elementos

echo ${#arreglo_b[@]} # cantidad de elementos
```

Comillas

- Tipos de comillas

- “Comillas dobles”:

```
var='variables'
echo "Permiten usar $var"
echo "Y resultados de comandos $(ls)"
```

- ‘Comillas simples’:

```
echo 'No permiten usar $var'
echo 'Tampoco resultados de comandos $(ls)'
```


Reemplazo de Comandos

- Permite utilizar la salida de un comando como si fuese una cadena de texto normal.
- Permite guardarlo en variables o utilizarlos directamente.
- Se la puede utilizar de dos formas, cada una con distintas reglas:

```
$(comando_valido)
`comando_valido`
```

Nota: La primer forma resulta más clara y posee reglas de anidamiento de comandos más sencillas.

- Ejemplo:

```
arch="$(ls)" # == arch="`ls`" == arch=`ls`
mis_archivos="$(ls /home/${whoami})"
```

Estructuras de Control

Ifs

Decisión:

```
if [ condition ]
then
block
fi
```

Selección:

```
case $variable in
"valor 1")
block
;;
"valor 2")
block
;;
*)
block
;;
esac
```

Iteración

- *C-style*:

```
for ((i=0; i < 10; i++))  
do  
block  
done
```

- Con lista de valores (foreach):

```
for i in value1 value2 value3 valueN;  
do  
block  
done
```

While

while

```
while [ condition ] #Mientras se cumpla la condición  
do  
block  
done
```

until

```
until [ condition ] #Mientras NO se cumpla la condición  
do  
block  
done
```

Operadores para condition:

Operador	Con strings	Con números
Igualdad	"\$nombre" = "Maria"	\$edad -eq 20
Desigualdad	"\$nombre" != "Maria"	\$edad -ne 20
Mayor	A > Z	5 -gt 20
Mayor o igual	A >= Z	5 -ge 20
Menor	A < Z	5 -lt 20
Menor o igual	A <= Z	5 -le 20

```
break [n] # corta n iteraciones
continue [n] # salta a n iteraciones
```

Argumentos

- Los *scripts* pueden recibir argumentos en su invocación.
- Para accederlos, se utilizan variables especiales:
 - \$0 contiene la invocación al script.
 - \$1, \$2, \$3, ... contienen cada uno de los argumentos.
 - \$# contiene la cantidad de argumentos recibidos.
 - \$* contiene la lista de todos los argumentos.
 - \$? contiene en todo momento el valor de retorno del último comando ejecutado.

Terminación de un Script

Para terminar un *script* usualmente se utiliza la función `exit`:

- Causa la terminación de un *script*
- Puede devolver cualquier valor entre 0 y 255:
 - El valor 0 indica que el script se ejecutó de forma exitosa
 - Un valor distinto indica un código de error

Funciones

Las funciones permiten modularizar el comportamiento de los *scripts*.

- Se pueden declarar de 2 formas:
 - `function nombre { block }`
 - `nombre() { block }`
- Con la sentencia `return` se retorna un valor entre 0 y 255
- El valor de retorno se puede evaluar mediante la variable `$?`
- Reciben argumentos en las variables `$1`, `$2`, etc.

Comandos Basicos

- `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`: 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.

Comandos importantes

let

Evalúa cada argumento como una operación aritmética.

```
let z=25; echo $z # Devuelve 25
let z++; echo $z # Devuelve 26
let z=z+10; echo $z # Devuelve 36
```

((dobles paréntesis))

Se comportan igual que la keyword “let”.

```
((z=25)); echo $z # Devuelve 25
((z++)); echo $z # Devuelve 26
((z=z+10)); echo $z # Devuelve 36
```

Si queremos retornar el resultado de una operación, usamos \$.

```
echo $((z=25)); # Devuelve 25
echo $((z++)); # Devuelve 26
echo $((z=z+10)); # Devuelve 36
```

test

Equivale a `[condición]`.

```
test 1 -eq 2 && echo "true" || echo "false"
# si es igual, se imprime true, si no, false

(test -f archivo && grep menta archivo && echo Z) || echo Q
# Si "archivo" existe y contiene el string "menta" imprime "Z", sino imprime "Q"
# Sin los paréntesis el resultado sería el mismo

test -w /home/pepe/
# evalua si el archivo o directorio existe y si tiene permiso de escritura
```

WC

Cuenta 🤖

```
wc -l <fichero> # número de líneas
wc -c <fichero> # número de bytes
wc -m <fichero> # número de caracteres
```

```
wc -L <fichero> # longitud de la línea más larga
wc -w <fichero> # número de palabras
```

cut

Corta ✂️

```
cut -d '.' -f1 archivo.csv # divide la linea con . y se queda con el 1er campo
cut -c 1-3 archivo.txt # se queda con los primeros 3 caracteres de cada linea
cut -c 1-3 --complement archivo.txt # complemento al anterior comando
# (todos menos los primeros 3)
cut -f 1,3 archivo.csv # se queda con los primeros 3 campos del csv
```

tr

Significa translate. Traduce 🐱

```
cat archivo.txt | tr [a-z] [A-Z]
car archivo.txt | tr [:lower:] [:upper:]
# traduce todas las minusculas a mayusculas
echo "hola soy taxi" | tr [:space:] '\t'
# traduce los espacios a tabs
echo "hola soy taxi" | tr -d x
# borra las x
echo "c8a4g0o3n" | tr -d [:digit:]
# elimina digitos
echo "c8a4g0o3n" | tr -cd [:digit:]
# complemento al flag -d -> elimina todo menos los digitos
echo "hola soy taxi" | tr ' ' '\t'
# traduce los espacios a tabs
```

grep

Busca texto en archivos 🔍

```
grep 'palabra' archivo.txt # busca 'palabra' en archivo
grep 'palabra' archivo1.txt archivo2.txt # busca en varios archivos
grep -i 'texto' archivo.txt # busca sin case sensitive
grep -w 'hola' archivo.txt # busca solo las palabras completas que coinciden
grep -r 'hola' directorio/ # busca recursivamente en un directorio
grep -v 'hola' archivo.txt # busca las que no coinciden
```

find

Busca 

find [RUTA] [OPCIONES] [PATRON]

```
find / -name archivo.txt # busca desde '/' por nombre 'archivo.txt'
find /home -name '*.jpeg' # busca por extension desde /home
find /ruta -type f # busca todos los archivos desde ruta
find /ruta -type d # busca todos los directorios desde ruta
```

cat

Concatena / Muestra

```
cat archivo.txt # muestra el contenido
cat archivo1.txt archivo2.txt # muestra el de los dos
cat -n archivo.txt # muestra con numero de lineas
cat archivo1.txt > archivo2.txt # copia destructivamente
cat archivo1.txt >> archivo2.txt # agrega
cat -s archivo.txt # suprime lineas en blanco
cat -E archivo.txt # muestra con un $ al final de cada linea
```

Comandos de Usuarios

- **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

Comandos de Permisos

chmod

Permite modificar y especificar quien puede manejar el archivo y cómo puede hacerlo.

Permiso	Valor	Octal
Lectura	R	4
Escritura	W	2
Ejecución	X	1

Pueden haber permisos de dueño o grupo.

Para cambiar los permisos, se utiliza el comando **chmod**, junto con la notación octal. Un ejemplo sería

```
$ chmod 755 /tmp/script
```

Donde se habilitan todos los permisos para el dueño con el primer dígito, para el grupo el segundo dígito y para todos los demás el último 5.

chown

Cambia al propietario del archivo

```
chown taci <archivos...>  
chown taci -R <directorio> # recursivamente
```