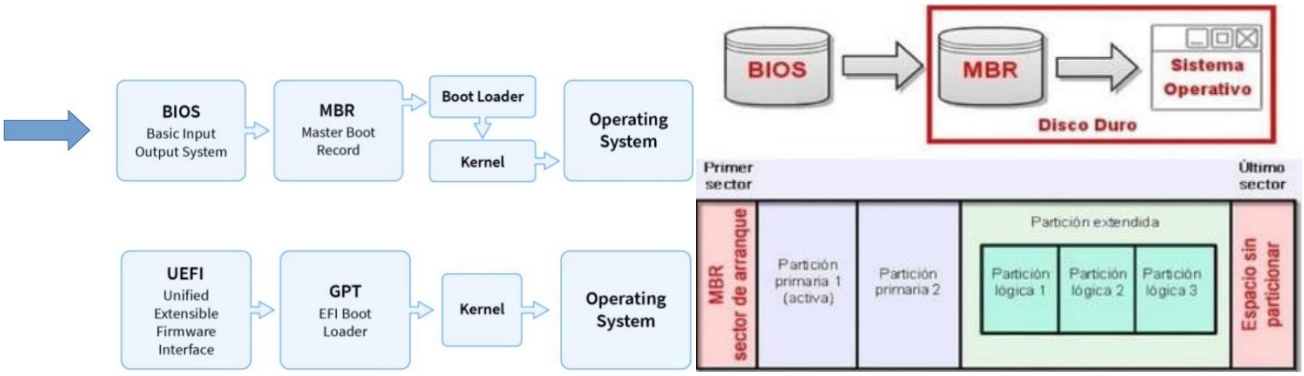
ROMPEMEGATON SISTEMAS OPERATIVOS

hay más de 1500 sistemas operativos, kernel no es un proceso.
Sistema operativo: programa que controla la ejecución de aplicaciones y programas y actúa como interfaz entre las aplicaciones y el hardware del computador.
La virtualización es una tecnología informática que simula la funcionalidad del hardware para crear servicios de TI basados en software.
Una máquina virtual es un entorno virtual que funciona como una computadora dentro de una computadora. Se ejecuta en una partición aislada con sus propios recursos de CPU, memoria, sistema operativo y otros recursos.
Un hipervisor (o monitor de MV), es un proceso que crea y ejecuta máquinas virtuales, y permite que un ordenador host preste soporte a varias máquinas virtuales invitadas mediante el uso compartido virtual de sus recursos.
Hipervisor tipo 1: llamado bare metal, se ejecuta sobre el hardware del host, en equipo sin SO. Ventajas: mejora rendimiento, mejor manejo de recursos, estabilidad. Desventajas: requiere soporte de hardware a nivel CPU, dedicación exclusiva de hardware, entornos empresariales, se necesitan 2 compus.
Hipervisor tipo 2: llamado alojados, se ejecuta como una capa de software sobre un SO de una pc física, ejecución similar a un programa/aplicación. Ventajas: prender/apagar MV a demanda, no necesita una re pc. Desventajas: recursos limitados, funciona bajo SO host asi q si falla cagamos, compite por los recursos con otros procesos, menor performance que las tipo 1.
Ventajas virtualización: aislamiento, seguridad, flexibilidad, agilidad, portabilidad, recuperación rápida en caso de fallo, ahorro de energía y dinero, administración centralizada.
Desventajas virtualización: necesidad de hardware de altas prestaciones (inversión grande), dependencia de un SO, limitaciones en el hardware de MV, problemas de compatibilidad entre algunos hipervisores.
Usos virtualización: probar SO nuevos, experimentar migraciones a SO nuevos, testear aplicaciones o migración de aplicaciones, mantener SO obsoletos con aplicaciones viejas, ejecutar archivos infectaos y probar comportamientos, crear copias de seguridad.
Requerimientos Linux Mint: 2 GB RAM, 20 GB en disco.
Comandos:

- **pwd**: imprime por pantalla el nombre del directorio de trabajo actual.
- **ls**: lista de contenidos de los directorios.
- **cd**: cambia el directorio de trabajo.
- **man**: muestra el manual de un comando (man page).
- **whoami**: devuelve cual es el usuario con el que estamos trabajando actualmente.
- **uname**: muestra información del sistema. (en la rta del comando te tira nros eso es la versión del kernel).

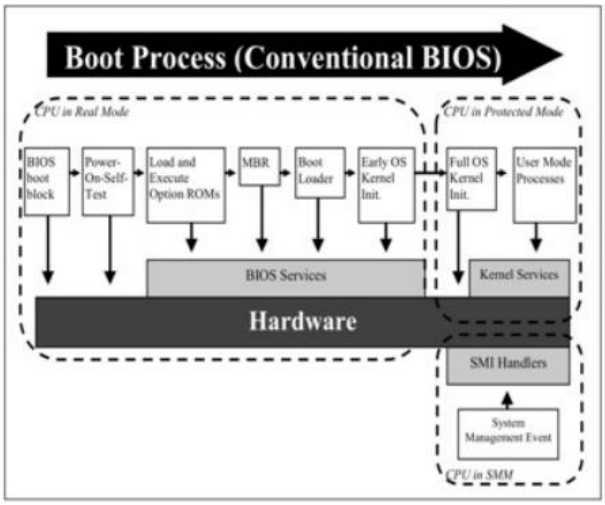
Arranque de una computadora personal con BIOS: prender la pc, lectura del contenido de la memoria BIOS, pruebas de los dispositivos del sistema POST, se muestra pantalla del estado, testeo de la memoria y otros recursos, búsqueda del primer dispositivo para cargar un sistema operativo.
ampliando la última etapa: antes de cargar un sist operativo, se busca un medio para iniciar el mismo (cd/dvd, usb, red), se leen los primeros 512 bytes de ese dispositivo para buscar un programa cargador del SO (MBR) y se selecciona un SO y se carga.

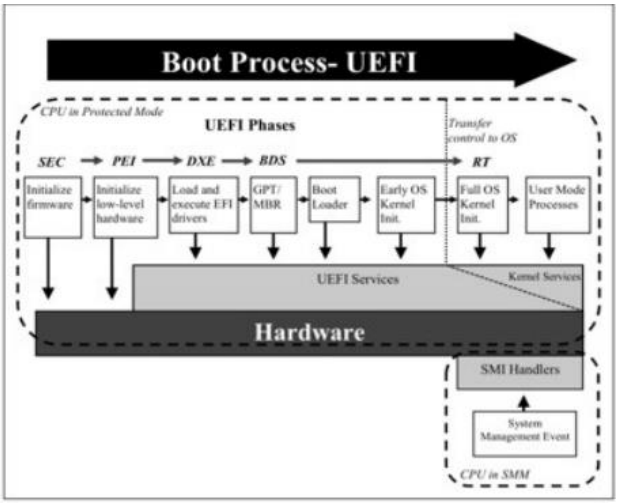


algunos gestores de arranque son: lilo, elilo, grub 2, bootmgr, etc.
Sistemas de inicio en GNU/Linux: sistema que controla la ejecución de los programas al inicio.
los tipos de sistemas de inicio son: BSD tradicional, System V, upstart, systemd (actual).
SYSTEM V:

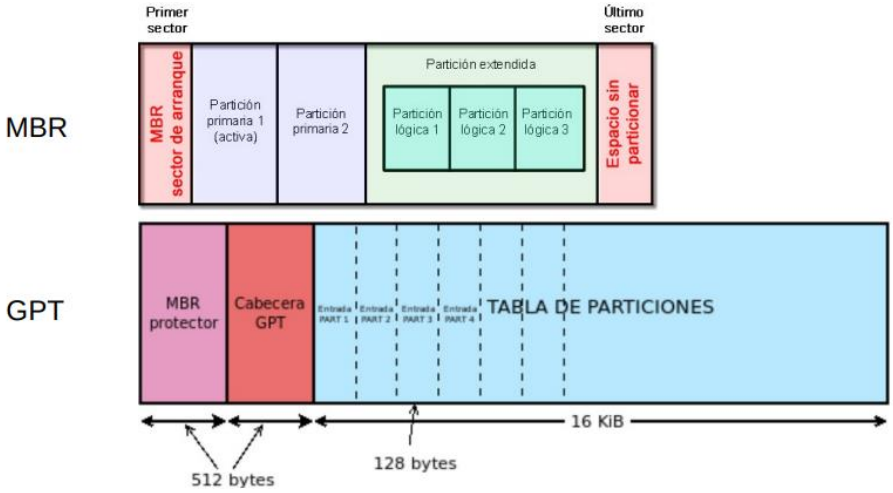
- inicio: el núcleo se descomprime y se carga en memoria para ser ejecutado, se inicializan diversos dispositivos de hardware, se monta el sistema de archivos principal (la partición raíz /), se ejecuta el proceso /sbin/init.
- runlevels: modos de ejecución de los SO basados en Unix, determinan formas de iniciar el SO y los servicios que podemos arrancar en cada uno de ellos, el demonio init se encarga de esto. 0 apagado del equipo, 1 inicio en modo mono-usuario, 2-5 inicio en modo multi-usuario y personalizables, 6 reinicio del equipo.
- script: guión o conjunto de instrucciones que permiten la automatización de tareas creando pequeñas utilidades, cada runlevel contiene scripts para iniciar, detener y reiniciar los diferentes demonios del sistema.
- demonio: tipo especial de proceso informático que se ejecuta en segundo plano en lugar de ser controlado directamente por el usuario, proceso no interactivo.
- getty: abre un puerto tty en el cual se ejecuta un proceso, mayormente el proceso login.
- login: ofrece un prompt para que el usuario ingrese al sistema y abre una Shell para que el mismo interactúe.
- init -> getty -> login -> Shell

en resumen: se enciende la compu, se ejecuta el POST, se busca dispositivo de arranque, se lee el primer sector del disco de memoria (primeros 512 bytes-MBR), se selecciona un SO y lo carga, descomprime el núcleo y se carga en memoria, se inicializan dispositivos de hardware (controladores), se monta el sistema de archivos raíz (“/”), se ejecuta el demonio /sbin/init como proceso padre de todos, el init se encarga de configurar e iniciar servidores del nivel, se inicia el proceso getty y se ejecuta el proceso login.





Como se puede ver en el gráfico, el BIOS ocupaba apróx menos de la mitad de la fase de booteo del pc, con uefi es distinto, ya que está presente prácticamente en el 90% del proceso. Incluso se puede apreciar que está unido al kernel.



La interfaz de firmware extensible y unificada UEFI: está diseñada para controlar el proceso de arranque y proporcionar una interfaz entre el firmware y el SO, a diferencia de BIOS, esta interfaz ofrece arquitectura propia, es independiente del proce y tiene sus drivers. UEFI no puede montar particiones ni leer algunos sistemas de archivos.
cuando un equipo tiene arranque UEFI, la interfaz busca una partición con identificador único global GUID que lo señala como partición del sistema EFI, dicha partición tiene aplicaciones compiladas que incluye gestores de arranque y que cuando se lo selecciona, UEFI le cede el control del proceso de arranque.
En el caso de grub, lo que se hace es ubicarlo en el sistema, y ejecutarlo directamente, cuyo nombre es grub.efi. GRUB se usa para elegir cual sistema operativo o kernel iniciar, cuando se lo selecciona, lo carga en memoria y le transfiere el control a esa máquina o sistema operativo.
sudo fdisk -l lista las particiones de un disco.
msconfig: servicio que se encarga de lanzar procesos al inicio en Windows, tiene múltiples pestañas para configurar cosas, la más útil es Servicios que muestra los demonios. (este lo clavas en el buscador y se abre System Configuration).
Administración de usuarios y grupos gnu/Linux: 2 a N usuarios trabajando en simultaneo, característica heredada de unix, usa ACL como mecanismo de control de acceso, define tipos de usuario para el sistema, todo usuario pertenece a un grupo determinado.
/etc/passwd: contiene información de los usuarios del sistema.
/etc/shadow: contiene información sobre la contraseña de los usuarios.
/etc/group: contiene información sobre los grupos del sistema.
Tipos de usuarios en gnu/Linux: usuarios especiales o del sistema, usuarios normales, súper usuario o root.

Comandos básicos para mostrar información de los usuarios:

- **who**: muestra información sobre los usuarios logueados en el sistema.
- **w**: muestra información sobre los usuarios logueados en el sistema y el comando que están ejecutando.
- **finger**: muestra información sobre los usuarios, su directorio /home, su shell por defecto y su nombre completo.

Comandos básicos para el manejo de usuario:

- **su**: Permite cambiar de un usuario a otro, se utiliza para cambiar a root o a súper usuario.
- **whoami**: devuelve cual es el usuario con el que estamos trabajando actualmente.
- **passwd**: cambiar nuestra contraseña o la de un usuario determinado.

Todo archivo posee un usuario y grupo determinados permite definir quién puede acceder a ese y de qué manera puede hacerlo, para ello se utilizan reglas de control de acceso (ACL), dependiendo del tipo de archivo se pueden especificar distintos tipos de permisos.

Maneras de administrar usuarios y grupos: editando archivos de configuración, por línea de comandos y/ o utilizando herramientas gráficas.
Por linea de comandos para usuarios: SINTAXIS *comandito <opciones usuario*

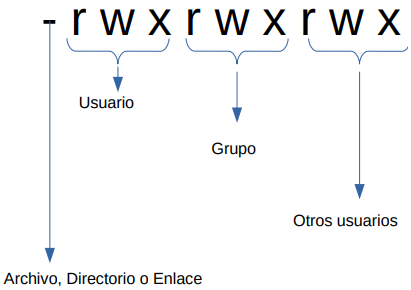
- **adduser**: agrega un usuario en el sistema, y las opciones son: --home especifica el directorio de inicio del usuario en el sistema, --shell especifica el intérprete de comandos del usuario, --no-create-home no crea el directorio de inicio del usuario.
- **deluser**: elimina un usuario del sistema, y las opciones son: --backup realiza un respaldo de los archivos que se encuentran en el home del usuario, --backup-to especifica el lugar donde se guardara el respaldo de la home del usuario, --remove-all-files remueve todos los archivos que sean propiedad del usuario a eliminar.
- **usermod**: modifica una cuenta de usuario del sistema, y las opciones son: -e especifica la fecha en la cual se desactivará la cuenta de usuario, -c modifica el valor del campo de comentarios del archivo passwd del usuario, -d modifica la ubicación del directorio de inicio del usuario.

Por linea de comandos para grupos: SINTAXIS *comandito <opciones> grupo*

- **addgroup**: agrega un grupo al sistema, y las opciones son: --version muestra por pantalla la versión del comando y copyright, --help muestra en pantalla ayuda del comando.
- **delgroup**: elimina un grupo del sistema.
- **groudmod**: modifica un grupo de usuario del sistema, y las opciones son: -n permite cambiar el nombre del grupo, -g permite cambiar el ID del grupo y debe ser un número entero positivo.

Permisos: Establecen la forma en la que un usuario, un grupo u otros usuarios pueden acceder a un recurso, usando **ls -l** se listan los archivos junto a los permisos que tienen.
tipos de permisos:

- **de lectura r**: archivo ve su contenido, directorio lista su contenido.
- **de escritura w**: archivo modifica su contenido, directorio borra y renombra.
- **de ejecución x**: archivo lo ejecuta, directorio poder acceder a él.



Administración de permisos por linea de comandos: SINTAXIS *comandito <permisos> archivo*

- **chmod**: modifica los permisos de un archivo, y las opciones son: -R permite que los permisos se apliquen a todos los subdirectorios de uno especificado, -v muestra información sobre los cambios realizados.
- **chown**: cambiar el dueño y/o grupo de un archivo, sintaxis *chown usuario(:grupo) archivo*. tiene las mismas opciones que el d arriba.
- **chgrp**: cambiar el grupo de un archivo, las opciones son las mismas d arriba.

en Windows podes ver usuarios y grupos poniendo en el buscador: **lusrmgr.msc**
el programa es distinto de proceso

gnu/Linux como sistema operativo multitarea:
Múltiples procesos pueden ejecutarse “simultáneamente” sin interferir entre sí, Cada proceso “cree” que se está ejecutando solo en el sistema y que tiene acceso a todos los recursos del mismo, Todos los procesos tienen siempre un determinado estado en el tiempo, El sistema operativo mantiene información de todos sus procesos y de su estado, Para cada proceso deben reservarse recursos y asignarse prioridades.

Init es el primer proceso a ejecutarse por el núcleo, tiene un identificador de proceso (PID) igual a 1, es el Padre de los demás procesos, realiza acciones y ejecuta varias instancias (forks) del proceso *Getty*.
Para enviar señales usamos el comando **kill.**

- **15 SIGTERM**: Solicita la terminación del proceso que la recibe.
- **2 SIGKILL**: Termina el proceso que la recibe en forma inmediata.
- **2 SIGINT**: Es la misma señal que se produce cuando un usuario presiona las teclas "Control + C" para solicitar su terminación.
- **20 SIGSTP**: Es la misma señal que se produce cuando un usuario presiona las teclas "Control + Z" para mandar un proceso a segundo plano.
- **19 SIGCONT**: Reanuda un proceso suspendido con SIGSTP.

Información sobre procesos:

- PID identificador de proceso.
- PPID identificador de proceso padre.
- valor de los registros.
- información sobre usuarios y grupos.
- prioridad con respecto a otros.
- recursos consumidos por el proceso.
- estado del proceso.

comandos para ver procesos:

- **ps** y **pstree**: muestra usuario que ejecuta el proceso, PID, % uso cpu y memoria, terminal, fecha inicio, comando que ejecuta.
- **top**: muestra la lista de procesos que consumen más recursos en el sistema, sintaxis *top <opciones>* y las opciones son: -s deshabilitad modo interactivo, -c muestra path absoluto con el q fue llamado cada comando de la lista, -d <segundos> especifica segundos de espera
- **htop**: muestra la lista de procesos y permite gestionarlos, F1 muestra ayuda, F permite ordenar procesos por consumo de memoria, cpu, usuario o pid, F10 sale de htop.

comandos para enviar señales a procesos:

- **kill**: enviar señales a un proceso, sintaxis *kill <opciones> <señal> PID* y las opciones son: -l muestra tabla con señales disponibles y respectivos números, -s señal especifica una señal por su nombre para enviar a un proceso, -n señal especifica señal por su número para enviar a un proceso.
- **killall**: permite enviar señales a un proceso usando su nombre, sintaxis *killall <opciones> <señal> nombre*, mismas opciones q arriba.
- **nice**: cambia la prioridad de un proceso a la hora de ejecutarlo, sintaxis: *nice <opciones> <proceso> argumento del proceso*, y las opciones son: -n permite especificar un número entero para definir la prioridad del proces, --version permite mostrar la versión actual del comando, --help muestra la ayuda del comando.
- **renice**: cambia la prioridad de un proceso en ejecución, sintaxis *renice prioridad <-p pid> <-g grupo> <-u usuario>*, y las opciones son: -p pid especifica el pid del proceso al que queremos modificar la prioridad, -g grupo cambia el id del grupo asociado al proceso, -u usuario cambia el id del usuario asociado al proceso.

más comandos para administrar procesos:

- **free**: muestra la utilización de memoria del sistema, sintaxis *free <opciones>*, y las opciones son: -m muestra la memoria en megabytes, -t muestra totales de memoria en las columnas de total, usada y libre, -s intervalo muestra el uso de la memoria y se actualiza pasado un intervalo de tiempo.
- **uptime**: indica el tiempo que lleva el sistema encendido, sintaxis *uptime <opciones>*, opciones: -v muestra la versión.
- **watch**: permite ejecutar un comando periódicamente, sintaxis *watch <opciones> comando*, y las opciones son: -n permite especificar un intervalo de tiempo para ejecutar un comando, -t permite no mostrar en la salida el comando que se utiliza, --help muestra la ayuda.
- **pgrep**: busca los procesos que tienen atributos determinados, sintaxis *pgrep <opciones> <patrón>*, y las opciones son: -G especifica nombre de un grupo para buscar los procesos, -l muestra la salida

del comando en formato largo agregando el nombre del proceso, -U intervalo especifica el nombre de un usuario para buscar procesos.

- **pstree**: muestra los procesos en forma de árbol, sintaxis *pstree <opciones>*, y las opciones son: -u muestra el nombre del usuario propietario del proceso, -p muestra pid del proceso.
- **fg, bg**: manipular y enviar procesos a primer y a segundo plano**,** sintaxis *fg, bg numero de trabajo*.
- **nohup**: ejecuta un comando y no permite que reciba más señales de hup, term y kill del sistema, sintaxis *nohup comando <argumentos>* y las opciones son: --version y –help.
- **&**: indica a la shell que un proceso se ejecute en segundo plano.
- **Jobs**: ver los procesos detenidos o en segundo plano en una Shell, sintaxis *Jobs <opciones>* y las opciones son: -l ver el pid de las tareas, -p muestra únicamente el pid de las tareas.

Hilos en POSIX

Llamada de hilo	Descripción
Pthread_create	Crea un nuevo hilo
Pthread_exit	Termina el hilo llamador
Pthread_join	Espera a que un hilo específico termine
Pthread_yield	Libera la CPU para dejar que otro hilo se ejecute
Pthread_attr_init	Crea e inicializa la estructura de atributos de un hilo
Pthread_attr_destroy	Elimina la estructura de atributos de un hilo

Para poder utilizar la interfaz de los hilos es necesario incluir la cabecera a pthread.h. Además, a la hora de compilar hay que enlazar el código con la opcion -pthread.
La creación de un hilo se hace mediante pthread_create. A partir de este punto, si la función no produce error, hay dos hilos de ejecución: el del programa invocante y otro cuyo nombre de función se pasa por parámetro y en nuestro caso se corresponde con primos_h. Dicha función recibe un puntero a datos y devuelve otro, en nuestro caso &tdata. Típicamente, el hilo invocante usa el ultimo parámetro de hilo para enviar datos de entrada al nuevo hilo.
El resultado del hilo se devuelve cuando la función con la que se crea el hilo finaliza. El hilo devuelve un puntero cuyos resultados se pueden recoger, más tarde, con pthread_join desde el hilo padre que lo ha creado.
Condición de carrera: Es cuando múltiples procesos o hilos leen y escriben datos de manera que el resultado final depende del orden de ejecución de las instrucciones en los múltiples procesos.
CRONTAB:

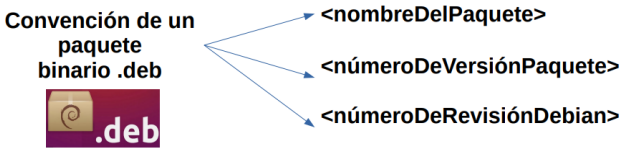
***(asterisco)** es una expresión de Cron, que representa “todo”, si lo pones quiere decir que la tarea se va a ejecutar cada minuto, cada hora, cada día de cada mes.
,(coma) se usa para separar por listas las programaciones. Por ejemplo: 0,30 * * * <comando> hace que se ejecute al principio y a la mitad de cada hora, todos los días, todos los meses.
-(guión) se usa para especificar un rango, 0-29 * * * * haría que el comando se ejecute del minuto 0 al 29 de cada dia etc etc etc
/(barra) se usa para expresar un valor de paso. Puedo hacer 0 */3 * * que significa que se ejecute cada 3 horas. Si no existiera, tendría que hacer un comando cada 3 horas (0 3,6,9,...,23 * *)

EJEMPLOS	DETALLE
* * * * * /home/aramburu/scripts/script.sh	cada minuto.
0 * * * * /home/aramburu/scripts/script.sh	cada hora, en el minuto 0.
0 4 * * * /home/aramburu/scripts/script.sh	diaria a las 4 de la mañana.
0 9,20 * * * /home/aramburu/scripts/script.sh	A las 9 de la mañana y a las 20 de la tarde (2 veces al día)
0 9-20 * * * /home/aramburu/scripts/script.sh	cada hora desde las 9 de la mañana hasta las 20 de la tarde.
0 9-20 * * 1-5 /home/aramburu/scripts/script.sh	cada hora entre las 9 de la mañana y las 20 de la tarde de lunes a viernes.
*/10 * * * * /home/aramburu/scripts/script.sh	cada 10 minutos.

comandos crontab:

- **crontab -l** lista los trabajos de cron.
- **crontab -e** edita o agrega una entrada a crontab.

Paquetes: distintas comunidades de usuarios desarrollaron Sistemas de Paquetes.



<nombre>_<NúmeroDeVersión>-<NúmeroDeRevisiónDebian>.deb

DPKG: herramienta para el manejo de paquetes de bajo nivel, similar al rpm, su función principal es la instalación, eliminación y configuración de paquetes Debian y su información. Para configurarlo se hace desde el archivo /etc/dpkg/dpkg.cfg

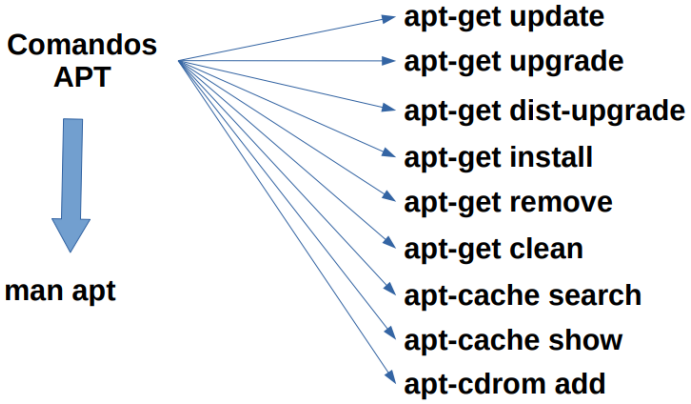
dpkg -i <paquete> Para instalar paquetes deb. No resuelve dependencias.

dpkg -r <paquete> Para desinstalar paquetes. No borra los archivos de configuración.

Gdebi es una herramienta que puede instalar paquetes deb. La cual permite instalar paquetes deb de forma local, es decir, sin necesidad de repositorios ni una conexión a Internet.

Sistemas de paquetes: es una colección de herramientas que sirven para automatizar el proceso de instalación, actualización, configuración y eliminación de paquetes de software. En estos sistemas, el software se distribuye en forma de paquetes, frecuentemente encapsulado en un solo fichero. Incluyen más info aparte del software como nombre completo, versión, descripción, etc. Esta información se introduce normalmente en una base de datos de paquetes local.

APT es una herramienta avanzada para el manejo de paquetes, basada en DPKG, para resolución de complejas dependencias, trae paquetes de servidores remotos para instalar, se realiza desde el archivo /etc/apt/sources.list (dicha dirección posee una lista de las fuentes en donde encontraremos los paquetes)



update sirve para ver la lista de paquetes disponibles para actualizar.
upgrade actualiza todos los paquetes instalados a sus últimas versiones, pero no agrega ni elimina nuevos paquetes que dependan de la actualización.
dist-upgrade es parecido, pero más agresivo, ya que este si puede agregar o eliminar paquetes.
clean limpia el cache de local de los paquetes descargados.
apt-cache search busca paquetes en la base de datos local.
apt-cache show muestra info de un paquete local.
apt-cdrom add suma un CD ROM con paquetes a las fuentes del software, para cuando instalas con CDs.

Existen herramientas gráficas para gestionar paquetes como **Kpackage, Shaman y Synaptic.**

KALI LINUX: distribución basada en Debian, enfocado en ciberseguridad y auditoría informática, desarrollado por Offensive Security. Surge como evolución de BackTrack (2006-2012) y en 2013 se convierte en Kali Linux. La motivación del SO es la necesidad de una plataforma más estructurada y funcional.
áreas principales:

- pruebas de seguridad: simula ataques para detectar vulnerabilidades en sistemas, redes y apps.
- auditoría informática: evalúa configuraciones y políticas de seguridad para identificar fallos o incumplimientos.
- análisis forense: recupera y analiza evidencias digitales sin alterar los datos originales a investigar.

Está enfocado a expertos o aprendices serios de ciberseguridad, no es recomendable a usuarios no familiarizados con Linux, sin conocimientos básicos en administración de sistemas, que buscan un Linux para aprender y familiarizarse, quiere una distro para usarla como SO habitual.
Tiene una actualización continua, no tiene soporte LTS (long term support), hay publicaciones oficiales 4 veces al año. Sus principales mejoras son el kernel actualizado, nuevas herramientas, correcciones de seguridad y optimización del sistema.
Las ramas de desarrollo: Kali-rolling, Kali-last-snapshot, Kali-dev / Kali-experimental.
Características:

- +600 herramientas preinstaladas.
- seguridad y autenticidad: acceso restringido a desarrolladores, firmas digitales en paquetes.
- compatible con muchas arquitecturas.
- modos de ejecución: Live CD/USB, HDD, máquinas virtuales.
- kernel con parches de inyección.

Categorías principales: recopilación de información, evaluación y explotación de vulnerabilidades, aplicaciones web, evaluación de bases de datos, ataques a contraseñas, ataques inalámbricos, intercepción y suplantación de identidad, post-explotación, herramienta de reporte, system services.
Políticas y seguridad: usuarios root, selección de herramientas, actualizaciones, servicios de red deshabilitados.
Gestión interna: usaba “root” como usuario por defecto ya que Kali era una distro pensada para auditorías, pruebas de penetración y tareas avanzadas, y muchas herramientas necesitan privilegios de admin. A partir de Kali Linux 2020.1 se dejó de usar para mejorar la seguridad, evitar malas prácticas entre usuarios principiantes y hacerlo más amigable para educación.
Se distribuye bajo licencias de software libre: GPLv3 (licencia publica general de GNU) y otras. No ofrece versiones de pago, es gratuito, libre y sin restricciones de uso.
Pros: acceso a ultimas herramientas, diseño especializado en ciberseguridad, actualizaciones constantes, compatibilidad con arquitecturas, flexibilidad en la ejecución y enfoque en software libre.
Contras: requiere conocimientos técnicos, actualizaciones continuas.