RESUMEN – PARCIAL 2 – ASG

Sistema operativo: Es un conjunto de programas que administran los recursos de hardware y software de una computadora.

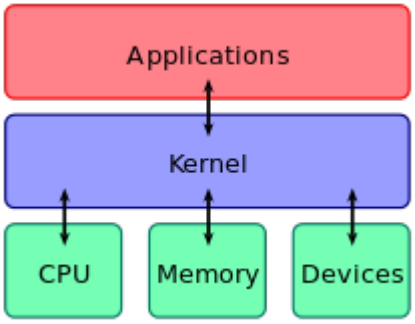
El Sistema Operativo (S.O.) controla y coordina el uso de recursos de Hardware y Software entre los diversos programas de aplicación.

Contraseña:

El sistema de contraseñas es de tipo unidireccional, lo cual significa que nuestra contraseña no es almacenada como texto, sino que es cifrada y guardada. Cuando entramos en el sistema y escribimos nuestra contraseña, ésta se cifra y se compara con la que hay almacenada.

Kernel del Sistema Operativo:

tiene el control de todo el sistema (Hardware y Software).
las aplicaciones solicitan servicios al S.O. a través de un mecanismo conocido como “llamadas al sistema” (system call).



Ejemplos de llamadas al sistema para que una aplicación escriba en un archivo:
open: se usa para obtener un descriptor de un archivo.
write: se escribe un dato en un archivo (dado su descriptor).
read: se usa para leer datos de un archivo (dado su descriptor).
close: cierra el descriptor de archivo.

La CPU ejecuta instrucciones de los programas. La ejecución será exitosa cuando:

- – Las instrucciones son comprendidas por la CPU (es decir, la CPU soporta el ISA que utiliza el programa).
- – Las llamadas al sistema (invocadas por medio de instrucciones del ISA) son comprendidas por el Kernel.

Un archivo es un recurso para registrar datos en un dispositivo de almacenamiento.

se utiliza un sistema de archivos que permite organizar los archivos en una cierta estructura, y controlar cómo se almacenan los datos en el dispositivo de almacenamiento y cómo se recuperan. Algunos sistemas de archivos son: FAT, NTFS, Ext2, Ext3, Ext4, APFS, etc.

los archivos están organizados en una estructura jerárquica, con directorios (o carpetas), que pueden contener archivos y otros directorios.

Estructura jerárquica de tipo árbol:

Administración de archivos en sistemas operativos tipo Unix:

Los sistemas operativos tipo Unix crean un sistema de archivos virtual, que hace que todos los archivos y directorios de todos los dispositivos parezcan existir en una sola jerarquía de tipo árbol.

El nivel más alto del sistema de archivos es / o directorio raíz. Todos los demás archivos y directorios están bajo el directorio raíz.

Cada "sub-árbol" dentro de la jerarquía puede residir físicamente en un diferente dispositivo de almacenamiento secundario local o remoto.

Todo esto es transparente al usuario, es decir, interactuamos con el sistema de archivos sin tener en cuenta dónde residen los archivos físicamente.

Los archivos se identifican únicamente por un nombre.

Los archivos no tienen extensión como en Windows. Sin embargo, si un programa (como el explorador de archivos) quiere saber de qué tipo es un archivo (imagen jpg, video mp4, documento pdf, etc.), puede resolverlo mediante un análisis simple de su contenido.

Un directorio no puede contener a la vez un directorio y un archivo con el mismo nombre.

Se distinguen letras mayúsculas de minúsculas.

Rutas:

Rutas absolutas: La ruta comienza con una barra "/" adelante, y los directorios se separan con barras "/"

Rutas relativas: La ruta NO comienza con barra "/". Los directorios se separan con barras "/". Para subir y bajar se utiliza ".." . Puede comenzar con "./" que significa "desde aquí".

EMPIEZAN LOS COMANDOS:

El shell puede ser escogido por el usuario: ¿Cuál estoy usando? `echo $SHELL`

`history` imprime los últimos comandos que hemos ejecutado y los numera. Para ejecutar uno de esos comandos, se pone ! seguido del número.

Para borrar la pantalla: `clear` borra lo visible.
`reset` borra todo (visible y anterior).

`Ctrl + c` detiene la ejecución de un comando.

`man` muestra información relacionada a un comando: `man nombre_comando`
Muestra:

- Nombre y descripción corta.
- Sinopsis (formato).
- Descripción del comando.
- Etc.

Para salir, presiona "q".

`man hier` describe cada uno de los directorios que aparecen creados en la jerarquía del sistema de archivos, una vez instalado el sistema operativo.

`echo $PATH` imprime las rutas de directorios en donde se encuentran los programas ejecutables.

Los comandos no son más que programas ejecutables que están en alguno de esos directorios.

Ruta absoluta del archivo ejecutable: `whereis comando`

COMANDO PARA MANIPULAR DIRECTORIOS:

Mostrar ruta absoluta del directorio actual: `pwd`
Cambiar directorio: `cd directorio`
Crear directorio: `mkdir directorio`
Borrar directorio vacío: `rmdir directorio`
Borrar directorio lleno: `rm -r directorio` . La opción "r" significa "recursivamente" , es decir, todo su contenido.
Para borrar todo el contenido del directorio actual: `rm -r *` El "*" significa "todo".

`ls` lista archivos y directorios:

- `ls -a` muestra archivos ocultos
- `ls -l` muestra más información
- las opciones se pueden combinar: `ls -l -a` o `ls -la`

`ls hola*.pdf` mostrará todos los archivos / directorios que sean pdf (en este caso), osea muestra todo lo que contenga los caracteres que hay desde el * hacia la derecha.

Copiar archivos: `cp origen destino` (son rutas (absolutas o relativas) a archivos o directorios.)
opción `-r` (antes de origen) para copiar recursivamente.

Eliminar archivos: `rm archivo` (En archivo hay que poner la ruta (absoluta o relativa) del archivo.)

Mover (y renombrar) archivos o directorios: `mv origen destino` (origen y destino son rutas (absolutas o relativas) a archivos o directorios.)

Imprimir contenido "en línea": `cat archivo`

Visualizar el archivo "fuera de línea": `less archivo`

Editar (o crear) archivo:
`nano archivo` (fácil y poco potente)
`vim archivo` (difícil y potente)

Determinar tipo de archivo: `file archivo`

Buscar archivos en todo el árbol de directorios: `locate archivo`

Buscar archivos en una rama específica del árbol: `find directorio -name archivo` (busca en cada directorio de la rama).

PARTICIONES:

Una partición de un disco es una división lógica de una unidad de almacenamiento (disco duro, pendrive, etc.), que puede ser administrada de manera independiente por el sistema operativo.

Existen distintos esquemas de particionado y los más conocidos son: MBR (Master Boot Record) y GPT (GUID Partition Table).

Ejemplos: FAT, NTFS, EXT2, EXT3, EXT4, APFS, etc.

Y AHORA… MAS COMANDOS.

¿que hace el KERNEL?

(¿no es eso lo que todos nos preguntamos? ¿que hace el kernel?)

Acceso a dispositivos desde el sistema de archivos:

En Linux los dispositivos se acceden desde el sistema de archivos:

`/dev` → contiene archivos que representan dispositivos. Ejemplos:

- `/dev/cdrom` es un lector de CD
- `/dev/sda` es un disco rígido
- `/dev/sdb` es otro disco rígido

Todos los discos tienen un prefijo que indica el tipo de conexión: ej: `sd` para disco SATA.
Las particiones se identifican con números.

Hay dos pseudo sistemas de archivos:

- `/proc` (proc filesystem) muestra información del kernel, hardware y procesos.
- `/sys` (sysfs filesystem) muestra información del kernel y hardware.

`lsblk -f` (list block devices) muestra información relacionada a los dispositivos de memoria secundaria del sistema.

`df` (`disk free`) muestra la capacidad de almacenamiento total, utilizada y libre.

`lscpu` : Muestra informacion sobre la CPU

`lstopo` (requiere paquete hwloc): lo muestra con una imagen muy bonita.

PERMISOS, tenes que saber sobre permisos:

definen quién puede acceder a un archivo/directorio y qué tipo de acceso puede realizar.

Tipos de acceso: lectura, escritura, ejecución.

Por cada archivo/directorio se otorgan permisos a: Dueño (owner), grupo (group) y otros (other).

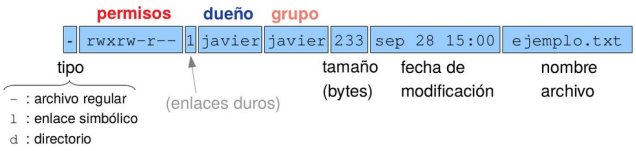
¿Quién puede cambiar los permisos de un archivo/directorio? El dueño o superusuario (root)

¿Quién puede cambiar el dueño y grupo dueño de un archivo/directorio?:
El superusuario (root), y el dueño podría cambiar el grupo dueño solo si él pertenece al nuevo grupo.

- **Lectura (r):** puede leer el contenido del archivo.
- **Escritura (w):** puede escribir el contenido del archivo.
- **Ejecución (x):** puede ejecutar el archivo.

Por ejemplo: con el comando `ls -l` podes ver informacion de un archivo o directorio:

un mambo asi:



Permisos de directorios (NO SE TOMA, PERO ESTA BUENO APRENDERLO, DALE APRENDE).

Lectura (r): puede listar su contenido. Es decir, ver una lista desus archivos y subdirectorios. Se requiere permiso de ejecución.

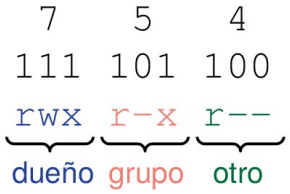
Escritura (w): puede modificar su contenido. Es decir, crear o eliminar archivos y subdirectorios (sin importar los permisos específicos que tenga sobre los archivos o subdirectorios). Se requiere permiso de ejecución.

Ejecución (x): puede ingresar al directorio. Es decir, puede operar con sus archivos y subdirectorios. Si no tiene permiso de Lectura para poder listar los archivos y subdirectorios, deberá conocer de antemano los nombres de los archivos y subdirectorios para poder operar con ellos.

OTRRRO COMANDO:

`chmod` = Comando que permite alterar los permisos de archivos y directorio.

```
Chmod 754 ejemplo.txt
```



r w x	octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Control de Dispositivos:

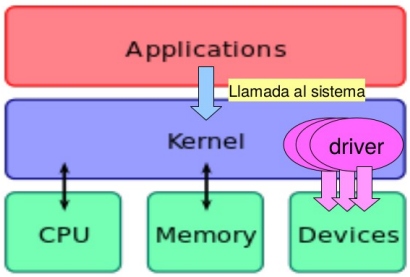
Por cada dispositivo de E/S soportado por un sistema operativo, alguien escribió un “driver” (controlador o mejor dicho manejedor).

Se necesita un controlador por cada dispositivo.

Si no está presente el controlador de un dispositivo, el Kernel no puede utilizar el dispositivo.

Un controlador/manejador de dispositivo es un programa informático que permite al kernel interactuar con un dispositivo de hardware.

Ejemplo: para mostrar algo en la pantalla, una aplicación ejecuta una llamada al sistema. A continuación, el kernel envía la solicitud a su controlador de pantalla, que es responsable de que los píxeles se visualicen en la pantalla.



Plug-and-Play ("enchufar y usar") es la tecnología que permite a un dispositivo ser conectado a una computadora sin tener que realizar configuraciones físicas (utilizando jumpers, interruptores, etc.) u otro tipo de intervención del usuario para resolver conflictos de recursos.

“Plug-and-Play” NO significa "no necesita drivers".

Los recursos pueden ser asignados en diferentes momentos:
– Tiempo de inicio/encendido de la computadora (boot time): cuando el dispositivo se encontraba conectado antes de encender la computadora.
– Una vez que el sistema ya ha iniciado: esto ocurre con los dispositivos que pueden conectarse y desconectarse en caliente (hotplug), es decir, mientras el sistema está en funcionamiento.

Los controladores generalmente ya vienen incluidos en el kernel Linux, el servidor de gráficos (Xorg) o impresión (CUPS). Si uno no está incluido, se puede descargar e instalar.

Administración de Procesos:

¿Qué es un proceso?

Es una instancia de un programa en ejecución. Mientras que un programa es una colección pasiva de instrucciones, un proceso es la ejecución real de esas instrucciones.

¿Cómo se crea un proceso?

Cuando el kernel quiere iniciar un programa, copia el programa desde el disco a la RAM. Para terminar de armar el proceso, el kernel le agrega información para su administración. A partir de este momento, el proceso queda disponible para ser ejecutado por el microprocesador.

¿Qué recursos de la computadora demanda un proceso?:

Memoria RAM: se requiere para almacenar el código y datos del programa (posiblemente mucho espacio) e información relevante para su administración por parte del kernel (poco espacio).

CPU: se requiere para ejecutar las instrucciones del código del proceso.

Multitarea:

¿Por qué una sola CPU (core) puede ejecutar múltiples procesos aparentando que ellos se ejecutan en paralelo (al mismo tiempo)?

El kernel del S.O. deja que la CPU ejecute instrucciones de un proceso durante un pequeño tiempo, y luego lo intercambia por otro proceso aunque el anterior no haya finalizado.

Estos intercambios son tan rápidos que el usuario cree que todos ejecutan en paralelo.

¿Cada cuánto tiempo mi kernel cambia de proceso?: Linux normalmente lo hace cada 100 ms.

¿Puede un proceso ejecutar por menos tiempo?: **PUES CLARO!**

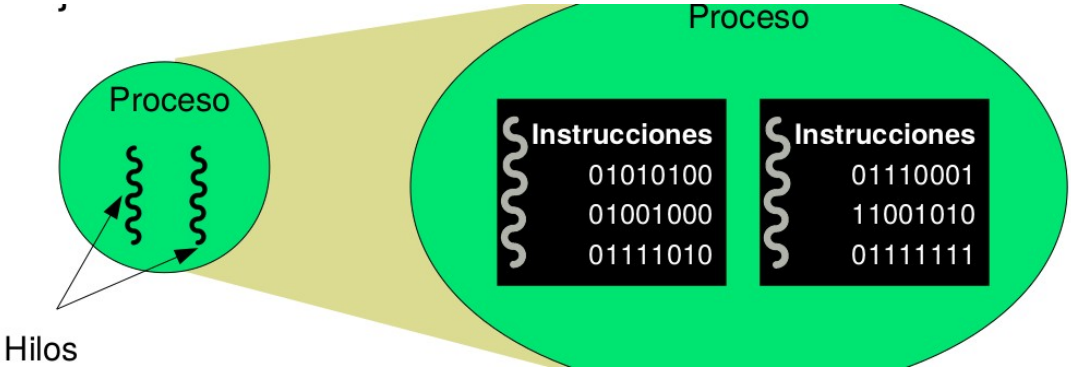
Pero…

Si, se da en dos situaciones:

- El proceso podría finalizar antes de terminar su ventana de tiempo.
- El proceso podría requerir que ocurra un evento para poder avanzar (por ejemplo, necesita datos del disco rígido o cualquier otro dispositivo). En este caso, se lo intercambia antes de terminar su ventana de tiempo para que la CPU continúe la ejecución de otros procesos sin pérdida de tiempo.

Hilos y programas multihilos:

Un hilo (thread) es una secuencia de instrucciones que puede ser gestionada de manera independiente. Un proceso tiene uno o más hilos de ejecución.



Los programas que utilizan más de un hilo se los conoce como multihilo y necesitan ser programados de una manera especial.

Todo proceso inicia con un solo hilo y luego se pueden ir creando nuevos hilos o eliminando.

¿Cómo se identifican los procesos y threads?:

Proceso: PID
thread: TID

El TID de uno de los threads es igual al PID del padre.

MAS Y MAS COOOMANDOS:

`ps` provee un listado estático de los procesos.
`ps -aux` muestra todos los procesos
`ps -T -p 819` muestra los threads del proceso con PID 819

`top` provee un listado dinámico de los procesos.

`kill` permite comunicarnos con los procesos, por ejemplo para indicarle que debe finalizar.

`kill -9 819` envía una señal de finalización al proceso con PID 819.

`pstree` permite ver la jerarquía de procesos.

Administración de Procesos en multicores:

Algunos microprocesadores multicore tienen CPUs que ejecutan instrucciones de un solo hilo. Sin embargo, hay multicores que Implementan la técnica de SMT (simultaneous multithreading), que permite a una CPU ejecutar diferentes hilos al mismo tiempo (incluso hilos de diferentes procesos).

Si un multicore tiene 4 cores, y cada uno puede ejecutar simultáneamente instrucciones de 2 hilos diferentes, se dice que el multicore tiene 8 cores lógicos.

Con SMT, los hilos comparten los recursos de la CPU, y esto es beneficioso cuando un solo hilo no puede aprovechar todo el poder de cómputo de una CPU.

En un computadora con un microprocesador multicore se ejecuta un único kernel de sistema operativo que administra todas la CPUs.

El kernel aprovecha las CPUs distribuyendo en ellas los hilos, del mismo proceso o de diferentes procesos, con el objetivo de acelerar la ejecución.

Administración de Memoria:

Un proceso tiene asignada una cierta cantidad de memoria RAM reservada solo para él. Los hilos del proceso utilizan la memoria asignada al proceso, es decir, la comparten.

Si el kernel del S.O. es multitarea, debe mantener múltiples procesos ejecutándose concurrentemente, y por lo tanto se necesita memoria principal para almacenarlos a todos.

El S.O asigna inicialmente una cierta cantidad de memoria a cada proceso.

Cuando el proceso finaliza, se libera el total de memoria asignada.

Durante su ejecución, un proceso puede solicitar al S.O. más memoria o liberar parte de su memoria ya asignada.

Memoria Virtual:

La memoria virtual es una técnica que crea la ilusión de tener una memoria RAM mayor a la físicamente disponible. Tiene dos ventajas:

- Permite ejecutar procesos que requieren un espacio de memoria de mayor tamaño que la RAM instalada en el sistema.
- Permite aumentar la cantidad de procesos, favoreciendo la multitarea.

La técnica consiste en reservar y utilizar una parte del disco para almacenar datos de procesos de manera temporal.

La desventaja es que el uso del disco hace al sistema muy lento.

Area de interacambio: memoria swap

Parte del espacio de memoria asignado a un proceso puede estar en la RAM y parte en el área de intercambio del disco.

Si se requiere memoria principal para ejecutar un proceso pero no hay más espacio libre, el S.O. debe hacer lugar. Para ello, mueve partes de procesos desde la memoria principal al área de intercambio.

Cuando ciertos datos que están en la memoria de intercambio necesitan ser procesados por la CPU, se mueven a la memoria principal.

Comando `free` | El comando top muestra la mismo info.

Lenguajes de programacion:

Los lenguajes de programación especifican reglas (como la sintaxis y semántica) para escribir programas.

Hay lenguajes de alto nivel y de bajo nivel.

Los de bajo nivel: (son re dificiles)

Sus instrucciones ejercen un control directo sobre el hardware. Las instrucciones son dependientes de la máquina.

Esta el lenguaje de maquina y el assembler *(lo que dice el capi cuando se pincha todo en end game) – Si sabes uno de esto, ya esta, Zeus te tiene que ceder el Olimpo -*

Lenguaje de maquina : Es el lenguaje que entiende la CPU, es decir, conformado por instrucciones del ISA.
Assembler: El lenguaje ensamblador utiliza:

- Mnemónicos en lugar de códigos binarios de operaciones. Ejemplo: ADD para indicar operación de suma.
- Identificación de registros en decimal: R1, R2, …
- Etiquetas en lugar de direcciones de memoria. Ejemplo: LOAD R1, cantidad
- Los datos numéricos se pueden escribir en decimal u otras bases. El texto se puede escribir utilizando caracteres.

Un lenguaje de bajo nivel da al programador un control total sobre lo que puede hacer con el procesador. (ESTOS LA LEVANTAN PERO EN PALA).

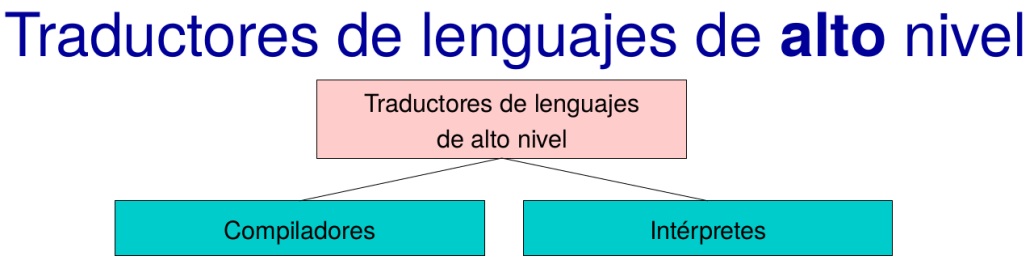
Son portables solo entre máquinas con igual ISA y S.O.

Lenguaje de alto nivel.

Están orientados a la resolución de diferentes tipos de problema.
Permiten especificar operaciones para resolver problemas en forma más parecida al lenguaje humano, matemático, etc.

Traductores:

El ensamblador es un traductor que convierte un programa escrito en lenguaje ensamblador a un programa escrito en lenguaje de máquina.

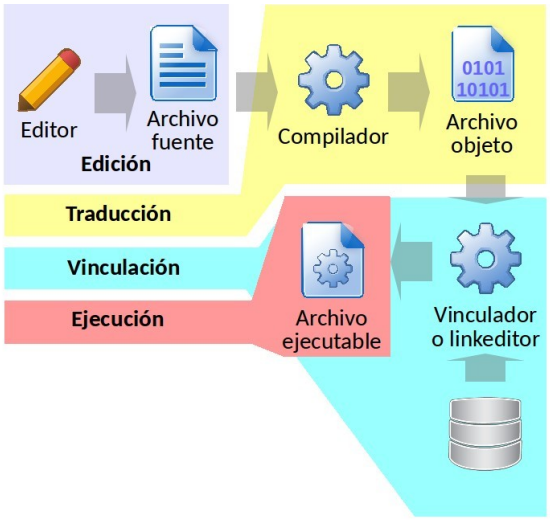


Los compiladores e intérpretes convierten código escrito en lenguaje de alto nivel a lenguaje de máquina.

Un compilador realiza la traducción (del código fuente) y almacena el resultado (código de máquina) en un archivo ejecutable. La traducción se hace una sola vez, y luego se ejecuta el programa tantas veces se quiera.

Ejemplo: los .exe de Windows.

Etapas del proceso de Compilación:



Edición del programa: el programador utiliza un editor para escribir el programa.

Traducción/Compilación: Se genera un código de objeto, que es un código de máquina que corresponde a la traducción del código fuente. Si se producen errores, el programador debe leer los mensajes de error y volver a la etapa 1 para corregirlos.

Vinculación (o enlazado): El programador, al escribir el código fuente, podría hacer uso de rutinas o funciones que vienen provistas por el lenguaje, y no necesita especificar cómo se realizan esas funciones. Al no aparecer en el programa fuente, esas funciones no aparecerán en el archivo objeto. La etapa de vinculación se ocupa justamente de completar el archivo objeto con el código de máquina faltante y logra así obtener el programa ejecutable.

Cuando un programa (escrito en un lenguaje interpretado) requiere ser ejecutado, el intérprete realiza la traducción de las instrucciones y las envía a ejecutar.

El código de máquina no se almacena en un archivo sino que directamente se envía a ejecutar. Al finalizar la ejecución del intérprete, habrá finalizado la ejecución de nuestro programa.

Traductores de lenguajes de alto nivel:

Intérprete:
– Es portable: el programa puede ser ejecutado directamente en cualquier computadora para la cual existe el programa intérprete (posiblemente con diferente hardware y sistema operativo).
– Bajo rendimiento: la CPU no está dedicada exclusivamente a ejecutar el programa sino también a la traducción del código fuente.

Compilador:
– No es portable: el programa solo puede ser ejecutado en computadoras con el mismo ISA y Sistema Operativo.
– Alto rendimiento: La ejecución del programa es rápida porque la traducción ya estaba hecha antes de ejecutarse el programa.

Instalación de Software:

Hay 3 opciones: vos elegí.

Archivos binarios: algunas aplicaciones se ofrecen directamente como programas ejecutables (código de máquina listo para ejecutar).

A partir del código fuente:

- Aplicaciones compilables: se compila el código fuente para producir un programa ejecutable.
- Aplicaciones interpretables: se ejecutan utilizando el intérprete que corresponda.

Paquetes de software (.deb, .rpm): las distribuciones preparan aplicaciones, las catalogan y las dejan listas para que el usuario las instale de manera simple.

WUW .DEB, ese uso yo.

¿Qué es una dependencia de software?
Algunos programas pueden depender de la presencia de ciertas librerías u otros programas para poder funcionar.

¿Qué es un paquete de software?
Conjunto de archivos agrupados que están relacionados a una aplicación de software.

Formato: cada distribución define su formato de paquete. Por ejemplo, Debian y derivados utilizan formato deb y RedHat y derivados utilizan rpm.
Contenido: los paquetes pueden tener binarios ejecutables, documentación, código fuente, etc.

Ventaja: es una forma simple, cómoda y rápida para instalar nuevas aplicaciones.

¿Qué es un Repositorio de Software?
Un repositorio de software es un lugar de almacenamiento del cual pueden obtenerse paquetes de software para instalar. Cada versión de cada distribución GNU/Linux tiene su propio repositorio.

Sistemas de Paquetes: gestores
Un gestor de paquetes de alto nivel permite buscar paquetes en los repositorios, instalarlos, actualizarlos y eliminarlos. Además, resuelven automáticamente las dependencias de software.

Gestos de comandos:

	Debian y derivados	RedHat y derivados
Tipo de paquete	deb	rpm
Gestor de bajo nivel	dpkg	rpm
Gestor de alto nivel	apt	yum

Un gestor de paquetes de bajo nivel instala un paquete de software que obtiene desde un archivo local. No resuelve dependencias.

```
sudo dpkg -i mi_paquete.deb
sudo rpm -i mi_paquete.rpm
```

Un gestor de paquetes de alto nivel permite buscar paquetes en los repositorios, instalarlos, actualizarlos y eliminarlos, y resuelve las dependencias.

```
sudo apt install mi_paquete
sudo yum install mi_paquete
```

¿Y si no tengo acceso de superusuario ?

Archivos binarios: ningún inconveniente.
A partir del código fuente: ningún inconveniente.
Paquetes de software: los gestores de paquetes de alto nivel requieren acceso de superusuario (root).

¿Qué puedo hacer?
– Usar un gestor de paquetes de bajo nivel, e indicar que el paquete va a ser instalado en un directorio propiedad del usuario (home). Habrá que instalar las dependencias manualmente.

¿Termino?