

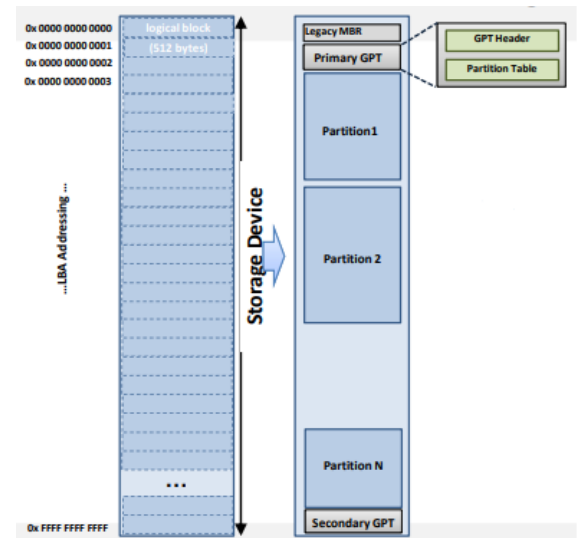
Esquema apuntes SI 2023: Samuel Castro Rodríguez

Tema 2: File Systems (Storage)

- Un sistema de ficheros es un mecanismo para organizar y representar el almacenamiento del sistema.

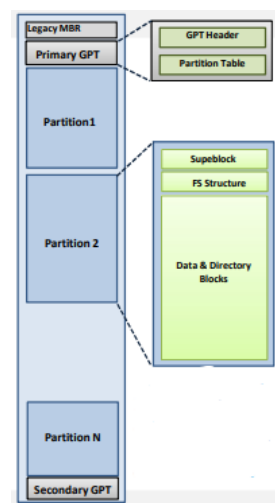
Elementos del sistema de almacenamiento:

- Partes de un disco:
 - **MBR**: primer sector del disco y contiene la tabla de particiones. También contiene un pequeño código de arranque llamado "bootloader" que se carga en la memoria y se ejecuta al encender el sistema.
 - **GPT**:
 - Formado por su cabecera y una tabla de particiones.
 - Las particiones de disco se definen en una sección del disco denominada GPT.
 - **Particiones**: Sección del disco utilizada para almacenar los datos.



Elementos de un Sistema de ficheros:

- **Dispositivo de almacenamiento**:
 - SSD
 - HDD
 - Flash Drive
 - ...
- **Partición** donde se alojará/definirá
 - Permite tratar a un único almacenamiento físico comportarse como si fuesen varios con un sistema de ficheros distinto para cada partición.
- **Sistema de ficheros**:
 - Es un componente del sistema operativo que actúa como mediador entre los bloques de datos crudos en el disco (raw data) y la interfaz estándar que los programas utilizan para acceder y manipular archivos. Proporciona una forma organizada y estructurada de almacenar y recuperar información en un disco.
 - El sistema de ficheros es como un traductor entre los programas y el disco duro de una computadora. Los programas utilizan nombres de archivos, ubicaciones y permisos para trabajar con la información, mientras que el disco duro solo sabe cómo almacenar datos en bloques secuenciales.
 - El sistema de ficheros se encarga de convertir lo que los programas quieren hacer con los archivos en instrucciones que el disco duro pueda entender. También organiza los archivos en carpetas y les asigna información adicional, como la fecha en que fueron creados y los permisos de acceso.
 - Para hacer esto, el sistema de archivos utiliza tres componentes principales:
 - Superbloque(Superblock), que contiene información sobre el sistema de ficheros en general.
 - Estructura del sistema de ficheros(FS structure), que define cómo están organizadas las carpetas y los archivos.
 - Datos en bruto(Raw data), que son los bloques de información reales almacenados en el disco.

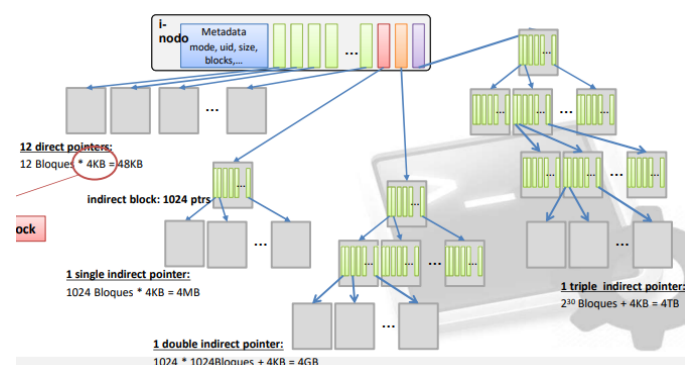


- **RESUMEN SISTEMA FICHEROS:** el sistema de ficheros es como un intermediario que ayuda a los programas y al disco duro a comunicarse de manera efectiva. Su trabajo es asegurarse de que los archivos estén organizados correctamente y tengan la información necesaria para ser utilizados por los programas.

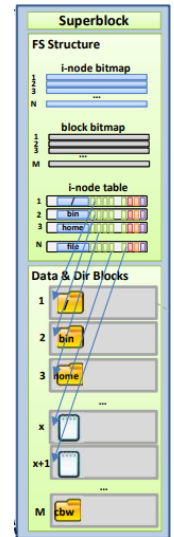
Sistema de Ficheros EXT:

- **i-nodo:** bloque básico sobre el que se construye el sistema básico de ficheros.
 - Almacena metadatos (info sobre los datos almacenados) y los bloques de datos en si.
 - Cada fichero o directorio tiene asociado un i-nodo.
 - Consumen un 10% del almacenamiento por defecto. (Se puede configurar en la creación del sistema de ficheros)
- En función de los punteros que tengo almacenados en el i-nodo determinare su **ocupación en memoria**. Se usa un sistema de punteros jerárquico.
 - Hay 4 tipos de **punteros de i-nodos en memoria**:
 - **Punteros Directos:**
Los punteros directos son los más simples. Cada i-nodo tiene una cantidad fija de punteros directos que apuntan directamente a los bloques de datos en el sistema de archivos. Por ejemplo, si un bloque de datos tiene un tamaño de 4 KB, un puntero directo puede apuntar a ese bloque y acceder a los datos almacenados en él.
→Es decir, si tenemos 12 punteros directos, con bloques de 4KB, el almacenamiento será de $12 \text{ bloques} * 4 \text{ KB} = 48 \text{ KB}$.
 - **Punteros de una indirección:**
Los punteros de una indirección permiten acceder a más bloques de datos. En lugar de apuntar directamente a los bloques de datos, los punteros de una indirección apuntan a bloques de punteros. Cada bloque de punteros contiene múltiples punteros directos, por lo que con un puntero de una indirección se puede acceder a más bloques de datos.
→Es decir, un bloque indirecto tiene 1024 punteros a bloques de datos de 4KB, siendo la ocupación en memoria de $1024 \text{ bloques} * 4 \text{ KB} = 4 \text{ MB}$.
 - **Punteros de doble indirección:**
Los punteros de doble indirección son similares a los punteros de una indirección, pero en lugar de apuntar a bloques de punteros, apuntan a bloques de punteros de una indirección. Cada bloque de punteros de una indirección a su vez contiene punteros directos a bloques de datos. Esto permite un acceso más eficiente a grandes cantidades de bloques de datos.
→Es decir, tengo un puntero que apunta a un bloque que hay 1024 punteros que apuntan a otros tantos bloques, así multiplicamos el número de bloques accesibles $\times 1024$, de tal manera que tendremos $1024 \text{ punteros} * 1024 \text{ bloques} * 4 \text{ KB} = 4 \text{ GB}$.
 - **Punteros de triple indirección:**
Los punteros de triple indirección son una extensión de los punteros de doble indirección. En lugar de apuntar a bloques de punteros de una indirección, apuntan a bloques de punteros de doble indirección. Cada bloque de punteros de doble indirección contiene punteros de una indirección, que a su vez apuntan a bloques de datos. Esto permite un acceso aún mayor a una gran cantidad de bloques de datos.

→Es decir, tenemos 1024 punteros que apuntan a otros 1024 punteros mas que a su vez apuntan a otros tantos bloques, de tal manera que tendremos $1024 \text{ punteros} * 1024 \text{ punteros} * 1024 \text{ bloques} * 4 \text{ KB} = 4 \text{ TB}$.



- Estructura del sistema de ficheros EXT (FS structure):
 - o **Bitmap de i-nodos:** mapa de bits que indica qué i-nodos están ocupados o libres.
 - o **Bitmap de bloques:** mapa de bits que indica qué bloques están ocupados o libres.
 - o **Tabla de i-nodos:** cada entrada es un único i-nodo.
 - Si mi bitmap de i-nodo no encuentra hueco pasa que no podre crear nuevos ficheros al no poder crear un i-nodo para este fichero.
 - ➔ Si un hacker crea ficheros vacios y ocupa todos los i-nodos disponibles, "infectará" tu ordenador no permitiendote crear mas ficheros.
 - Si mi bitmap de bloques no encuentra bloques disponibles no podré crear nuevos ficheros.
 - ➔ Ocupando todos los bloques de datos mediante la creación de un fichero de gran contenido.



- Tipos de Sistemas de Ficheros:
 - o **EXT 1-3 vs EXTENT:**
 - **Ext1:** El sistema de archivos ext1 es la primera versión del sistema de archivos ext utilizado en Linux. En ext1, la asignación de espacio en disco para un archivo se realiza mediante bloques individuales. Cada bloque puede contener una parte del archivo, y se utilizan punteros para rastrear y acceder a los bloques de datos.
 - **Ext2:** El sistema de archivos ext2 es la segunda versión del sistema de archivos ext. Al igual que en ext1, en ext2 se utiliza una asignación de bloques individuales para los archivos. Cada bloque de datos contiene una parte del archivo y se utiliza un sistema de punteros para acceder a ellos.

La fragmentacion de datos es un problema, los i-nodos y sus datos asociados pueden llegar a estar muy lejos. Para mejorar esto se intento acercar los datos a los metadatos, se replicaron los metadatos por grupo para que esten mas cerca (Mejoró la localidad de datos y metadatos).
 - **Ext3:** El sistema de archivos ext3 es una extensión del ext2 con la adición de una característica llamada "**journaling**" para mayor confiabilidad y recuperación en caso de fallos del sistema. Al igual que en ext2, en ext3 se utiliza una asignación de bloques individuales para los archivos, sin el uso de extents.
 - **Ext4:** En contraste, el concepto de "**extents**" se introdujo en el sistema de archivos **ext4**, que es la cuarta versión del sistema de archivos ext. **Ext4** mejora la eficiencia de almacenamiento al permitir la asignación de rangos contiguos de bloques para un archivo en lugar de utilizar bloques individuales. Esto reduce la sobrecarga de almacenamiento y mejora el rendimiento de acceso a archivos grandes reduciendo la fragmentación externa. También implementa journaling.

El **journaling**, también conocido como registro de transacciones, es una técnica utilizada en los sistemas de archivos para garantizar la integridad y la consistencia de los datos en caso de fallos del sistema, como cortes de energía inesperados o reinicios.

Cuando se utiliza el journaling, se crea un registro (journal) que actúa como un diario de transacciones. Antes de realizar cualquier modificación en el sistema de archivos, ya sea para escribir nuevos datos o modificar metadatos (información sobre los archivos, como nombres, ubicaciones, permisos, etc.), se registra la intención de la modificación en el journal.

El journal almacena información detallada sobre las transacciones pendientes, incluyendo los cambios que se realizarán y su ubicación en el sistema de archivos. Estos registros se escriben de manera rápida y eficiente en un área reservada del disco.

Cuando ocurre un fallo del sistema, como un corte de energía, el sistema de archivos puede recuperarse utilizando la información del journal. Durante el reinicio, el sistema verifica el journal y aplica o deshace las transacciones registradas en el journal según corresponda. Esto garantiza que el sistema de archivos vuelva a un estado coherente y evita la corrupción de datos.

RESUMEN JOURNALING: técnica que utiliza un registro de transacciones para asegurar que las modificaciones en el sistema de archivos se realicen de manera segura y que se pueda recuperar de manera confiable en caso de fallos del sistema. Proporciona mayor integridad y consistencia de los datos, reduciendo el riesgo de pérdida o corrupción de información.

Un **"extent"** es un concepto utilizado en los sistemas de archivos para gestionar y organizar el almacenamiento de archivos de manera eficiente. Representa un bloque contiguo de espacio en disco asignado a un archivo. En lugar de utilizar una lista de bloques individuales para almacenar un archivo, un extent permite almacenar una secuencia continua de bloques de datos.

La utilización de extents tiene varias ventajas. En primer lugar, reduce la fragmentación externa del disco, ya que los bloques de un archivo se almacenan de forma contigua, evitando la dispersión de los datos por todo el disco. Esto mejora el rendimiento del sistema de archivos, ya que la lectura y escritura de datos consecutivos se pueden realizar de manera más eficiente.

Además, el uso de extents reduce la sobrecarga de almacenamiento y mejora la eficiencia del espacio en disco. En lugar de mantener un puntero individual para cada bloque de datos, solo se necesita un puntero para cada extent, lo que ahorra espacio en la estructura del sistema de archivos y acelera la navegación y manipulación de archivos grandes.

Por otra parte, los extents pueden producir fragmentación interna cuando se modifican o eliminan datos de los bloques pudiendo llegar a dejar bloques intermedios vacíos.

Sistema de ficheros Raíz:

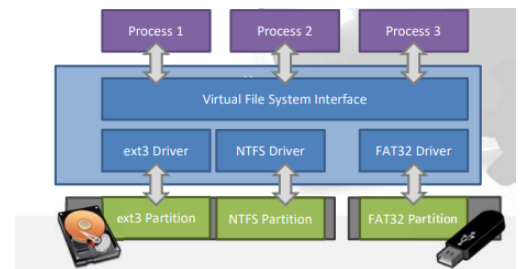
- El sistema de ficheros raíz es imprescindible que este disponible en arranque, ya que es el principal directorio sobre el cual acceder a todos los demás.
- **Enlace de tipo duro:** Un nuevo fichero que apunta al mismo i-nodo. Si yo borro el fichero original yo sigo teniendo acceso al contenido original por el enlace duro que cree antes, manteniendo vivo el i-nodo.
- **Enlace de tipo blando:** redirección, una especie de alias con el path para apuntar al fichero directamente, si yo borro el fichero el i-nodo se borra y no seguiría teniendo el contenido.
- Una **"Access Control List" (FYI)**, en español "Lista de Control de Acceso", es un mecanismo utilizado en sistemas operativos y sistemas de archivos para controlar y gestionar los permisos de acceso a recursos, como archivos, directorios o dispositivos.
 - o Es una lista asociada a un recurso que especifica los usuarios, grupos u otros sujetos y los permisos que tienen sobre ese recurso.
 - o Cada entrada en la lista de control de acceso contiene información sobre el sujeto y los permisos que se le otorgan, como leer, escribir, ejecutar o modificar.
- Si tengo un **fichero con el atributo "i"** que significa que no se puede borrar, tendría que cambiarle el atributo con los comandos **lsattr** y **chattr** a otro distinto para que pudiese ser borrado.

Block Device Naming:

- El nombramiento de mis **unidades de almacenamiento** será a partir de "**sd<letra>**" por ejemplo sda, luego están definidas **las particiones** con números tal que **sda1**. Las siguientes unidades de memoria se nombraran con las siguientes letras del abecedario como "sdb", y así sucesivamente.
- Es **importante mantener la consistencia en el nombramiento** de mis unidades de memoria y particiones, para esto tenemos que ser conscientes que cada unidad de almacenamiento y partición tienen un identificador único que siempre es el mismo y que los diferencia del resto, el **UUID**.
 - o Es necesario tenerlo en cuenta ya que al reiniciar el sistema o realizar cualquier acción sobre el almacenamiento, los nombres inicialmente asociados a las unidades de memoria y particiones pueden cambiar, creando así inconsistencias en las posibles acciones a tomar sobre ellos, como por ejemplo al realizar un montaje permanente en el fichero /etc/fstab, si definimos el montaje con sd<letra><numero> podremos tener inconsistencias así que lo mejor es utilizar el UUID de la partición a montar.

Managing Multiple Filesystems:

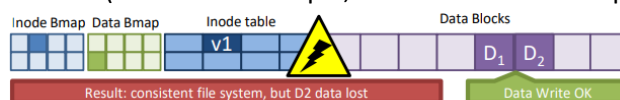
- El sistema operativo puede montar varias particiones con diferentes sistemas de ficheros asociados a cada una. → ¿Es necesario que un proceso utilice una API diferente para cada sistema de ficheros?:
 - o Linux hace un uso de una interfaz llamada **Virtual File System (VFS)**.
 - Expone una **POSIX API** a los procesos para que interactúe cada uno con los diferentes sistemas de ficheros. (Llamadas al sistema y operaciones estandarizadas para todos los sistemas de ficheros).
 - Reenvía las peticiones al driver específico del sistema de ficheros sobre el que actuar.



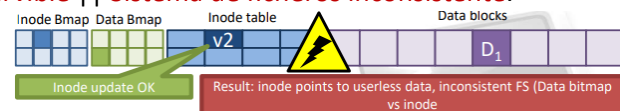
Consistencia del Sistema de Ficheros:

- Sistema de ficheros EXT: → ext3
 - o Algunas operaciones requieren múltiples e independientes operaciones de escrituras en el sistema de ficheros, como por ejemplo añadir un bloque a un fichero existente.
 - o ¿Qué pasa **si se interrumpe la operación** en un punto intermedio antes de que sea realizada?
 - Si la máquina en la que se está creando un fichero se apaga abruptamente sin cerrar bien todos los procesos, pueden pasar 3 cosas:

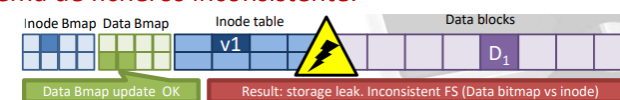
1. **Escritura de datos correcta** || **Pérdida de bloque de datos** || **Sistema de ficheros consistente**. (Escritura en bloque, no escritura en bitmap de inodos ni bloques).



2. **Actualización del i-nodo correcta** || **El i-nodo apunta a un bloque de datos inservible** || **Sistema de ficheros inconsistente**.



3. **Actualización del Data Bitmap correcta** || **Desperdicio de almacenamiento** || **Sistema de ficheros inconsistente**.



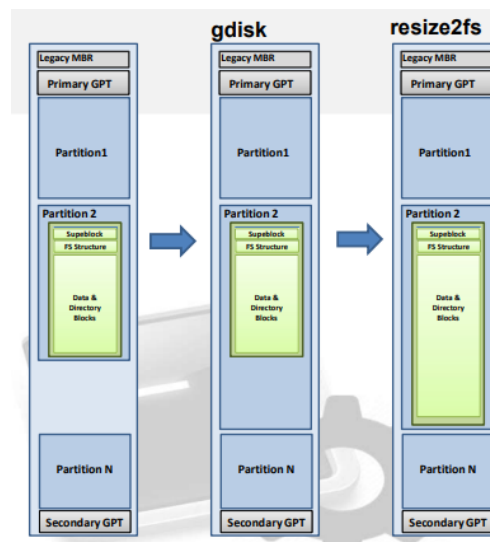
- Este tipo de problemas se intenta **mitigar con journaling**: todos los datos asociados a una modificación se escriben de manera simultánea en el disco.
 - Agrupado en el journal, donde estarán presentes todos los cambios.
 - Primero se escribe en el journal, y luego ya se pasa a escribir en el sistema de ficheros.
 - Como se tienen que escribir la información 2 veces, una en el journal y otra en el sistema de ficheros, se implementan 2 mejoras para optimizar el rendimiento:
 - ➔ **Almacenando en búfer las escrituras secuenciales en memoria**, agrupándolas como un solo registro. En lugar de escribir cada cambio de datos en el sistema de archivos de forma inmediata, se guarda temporalmente en un búfer en memoria. Luego, los cambios se agrupan y se escriben como un solo registro en el sistema de ficheros en un momento más oportuno. Esto permite reducir la cantidad de operaciones de escritura individuales.
 - ➔ **Realizando el registro solo para los metadatos** (mapa de bits de datos + i-nodo).
 - De esta manera, optimizamos el rendimiento al aplicar journaling.
 - ¿Qué pasa si la escritura en el journal es interrumpida?:
 - El sistema de ficheros permanece consistente pero no se ejecuta la transacción (pérdida de datos).
 - ¿Qué pasa si el journal es escrito correctamente pero los datos en el disco no?
 - El sistema de ficheros permanece inconsistente temporalmente.
 - Durante el proceso de boot(encendido) se reestablecen los journals que no estén completos.

Comando dd:

- Es capaz de leer y escribir en ficheros especiales, por lo que es capaz de copiar particiones completas, además de toda la estructura del sistema de ficheros, no como el cp que no copia esta estructura.

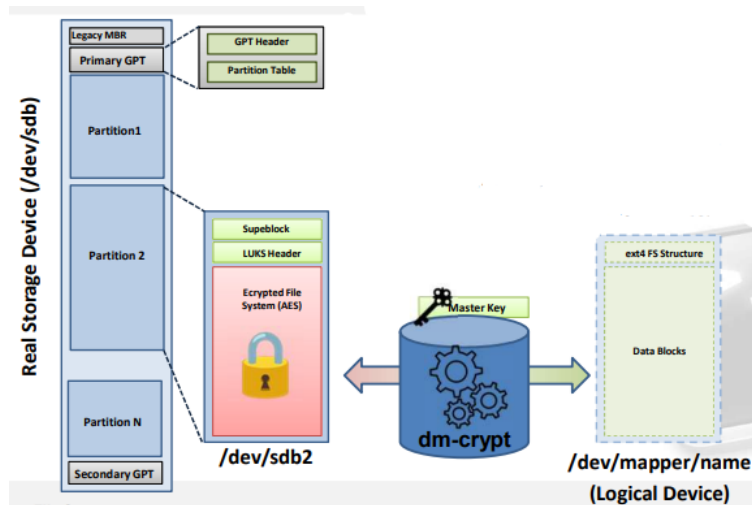
Redimensionar el Sistema de Ficheros:

- Pasos:
 - Mediante **gdisk**, crear la partición de mayor tamaño, respetando el primer sector del bloque que antes era de menor tamaño, y aumentar el tamaño del segundo sector hacia el que se acabará el bloque de memoria nuevo. De esta manera extendiendo el tamaño.
 - Posteriormente se asigna el tamaño a mi sistema de ficheros con el comando **resize2fs**.
- Cuando tengo una partición, lo que se espera encontrar arriba es el superbloque, si yo muevo el sistema de ficheros por la dirección de arriba, no encontraré el superbloque, es por esto que siempre se añade tamaño a las particiones hacia abajo.



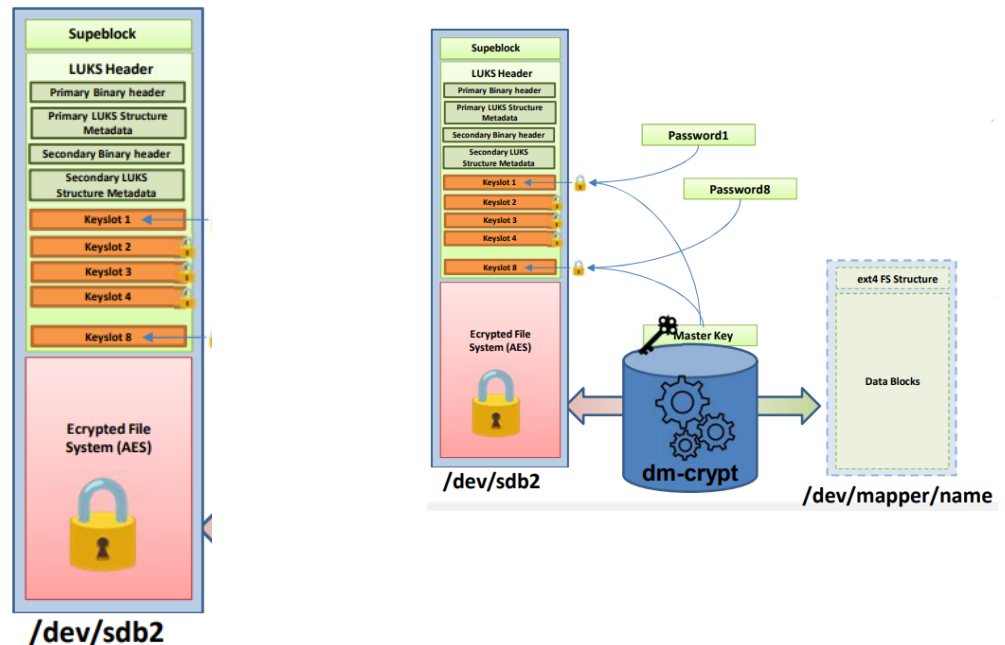
Encriptación: LUKS + dm-crypt

- **LUKS (Linux Unified Key Setup):** Sistema de cifrado de disco para particiones completas o dispositivos de almacenamiento.
 - o Es un sistema de cifrado de disco utilizado en sistemas operativos Linux.
 - o Proporciona una capa de cifrado a nivel de bloque para proteger el contenido de particiones completas o dispositivos de almacenamiento, como discos duros o unidades USB.
 - o Con LUKS, es posible cifrar toda una partición o dispositivo de almacenamiento de manera transparente, lo que significa que los datos se cifran y descifran automáticamente a medida que se leen y escriben en el disco.
 - Esto garantiza que los datos almacenados en el disco estén protegidos en caso de pérdida o robo físico del dispositivo.
 - o El cifrado de disco con LUKS se basa en el uso de una o más claves de cifrado. Estas claves se utilizan para cifrar y descifrar los datos, y están protegidas mediante una contraseña o frase de paso.
 - o Además, LUKS también admite el uso de claves de cifrado adicionales, como tarjetas inteligentes o tokens USB, para una mayor seguridad.
- LUKS encripta normalmente mediante el **mecanismo AES**, realiza una encriptación simétrica, significa que hay una única clave que es la que se usa para encriptar y desencriptar (**master key**).
- Necesito una **entidad traductora (dm-crypt)**, se basa en un driver que escriba encriptado todo el contenido que yo introduzco en esa partición y que por lo tanto para traducir ese contenido necesitaría la master key.
 - o Desde el punto de vista de seguridad si alguien consigue esa master key habría una vulnerabilidad grande, esta master key esta presente en el LUKS header, a la cual solo se puede acceder con permisos de administrador.
 - Como se almacena en el luks header, si consigo la estructura del luks header mediante el comando dd, podría obtener la master key y vulnerar la encriptación de determinadas particiones.

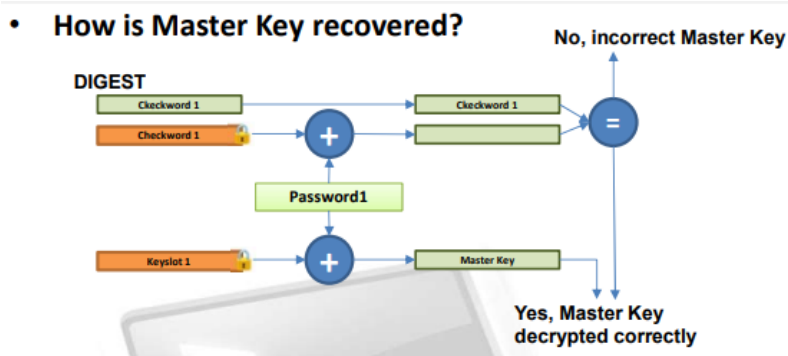


- **LUKS Header:**
 - o Si pierdo los datos de la cabecera, pierdo todos mis datos del fichero encriptado, ya que perdería la master key y no podría conocer el contenido real de la partición al estar encriptada toda la información.
 - o Está formado por:
 - Encabezado binario: Firma para detectar dispositivos LUKS, información básica, tamaño del encabezado, suma de verificación de metadatos.
 - Metadatos LUKS:
 - ➔ **Keyslots**: describen las áreas de almacenamiento de claves cifradas.
 - ➔ **Digest**: texto utilizado para verificar que las claves descifradas de los slots de claves sean correctas.
 - ➔ **Segmentos**: describen áreas en el disco que contienen datos cifrados.

- Encabezado/metadatos secundarios: copias de seguridad para evitar la corrupción de datos en el encabezado LUKS.
- KeySlots: parte del encabezado LUKS que almacena la clave maestra cifrada.
 - ➔ Se pueden definir múltiples contraseñas para un único disco cifrado.



- El **uso de digest** en una partición LUKS es importante para garantizar la integridad de las claves y detectar posibles manipulaciones o daños en ellas.
 - Es un tipo de función hash criptográfica que produce un valor único de longitud fija a partir de un conjunto de datos.
 - Si se detecta que una clave ha sido alterada o está dañada, se considera una señal de que la integridad de los datos cifrados puede estar comprometida.
 - Los digest se usan para recuperar la master key, sin tener que guardarla como texto plano.



- ➔ Cuando se accede a una partición LUKS, el sistema verifica el digest calculado del encabezado con el digest almacenado previamente. Si los dos coinciden, se considera que el encabezado no ha sido modificado y se puede proceder a desbloquear la partición y acceder a los datos encriptados.

Tema 3: User Management

Definición y Creación:

- Los usuarios se definen y crean en el fichero **/etc/passwd**.
 - o Esta definición se realiza mediante una nomenclatura específica en texto plano:

test:x:500: 500: Usuario test: /home/test: /bin/bash:

- **Name:** all the resources employed by the user identified with that label. Must be unique.
 - **Password:** the x indicates the account is protected by an encrypted password stored in the file with restricted access **/etc/shadow**.
 - **UID:GID:** numerical identifiers for user and group. UID must be unique. Coherent assignation policies: UID>999 (lower ids for system accounts). Do not reuse UIDs (¿What happens if we want to restore an old user?)
 - **GECOS:** Personal information about the user. Limitless, comma separated.
 - **Root directory:** system directory where the user stores its files.
 - **Shell:** binary associated to the shell employed by the user.
- o Que el UID sea < 1000, el 0 admin y entre el 0 y 1000 corresponde a procesos o a servicios del sistema, para gestionar el nivel de acceso y permisos a directorios y ficheros del sistema.
 - o Que el UID sea > 1000 corresponde a la declaración de nuevos usuarios (personas).
 - El editor de texto/comando **"vipw"** comprueba la sintaxis de lo escrito en el fichero.

- UNIX permite el manejo de los usuarios mediante grupos, definidos en el fichero **/etc/group**.

test:*: 500:

cdrom:*: 24: test

- **Name:** group name
- **Password:** in general, not employed.
- **GID:** group identifier.
- **Additional users:** a user can belong to more than one group.

- Además de crear en el directorio **/home** la carpeta correspondiente al nuevo usuario, hay que dotar a ese directorio de los permisos necesarios para que el usuario pueda acceder y modificarlo. Además de asignarle su grupo.
- Si hay que crear múltiples usuarios de una, lo mejor es automatizar la tarea mediante un script utilizando esta serie de comandos:
 1. Adduser
 2. Usermod
 3. Addgroup
- Existen varias maneras de inhabilitar o eliminar a usuarios:
 - o Cambiar la Shell a **/bin/false**:
 - No va a permitir acceder al sistema mediante la shell al usuario, pero si que podría acceder a los archivos si tiene un protocolo ftp de traspaso de archivos.
 - o Bloqueo de la cuenta:
 - Mediante el comando **passwd -l** "caducará" la contraseña del usuario.
 - o Eliminación de la cuenta:
 - Mediante el comando **userdel -r username** (-r elimina todo el contenido de su directorio **/home**).

Seguridad:

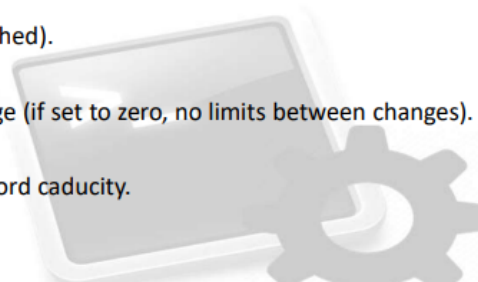
- Si recomendamos a los usuarios que no escriban sus contraseñas en texto plano, nosotros en el sistema ¿Qué implementaremos para cumplir con la seguridad?:
 - o **Algoritmos de encriptacion de una direccion**, mediante la llave accedo a la informacion descriptada, pero mediante la informacion encriptada no accedo a la llave.



- Mediante el comando **passwd** le asignamos al usuario una contraseña, que será guardada en el fichero **/etc/shadow** encriptada.
- Para que el usuario tenga acceso solo a su contraseña de su usuario y no a los demas usuarios, le concedo solo permisos al comando passwd y no al fichero /etc/shadow, ya que si cambia el hash de su contraseña en la de otro usuario podría acceder a él con su contraseña o incluso simplemente eliminar la contraseña del usuario.
- Si algun atacante tiene mi contenido de /etc/shadow y /etc/passwd, mediante un ataque de fuerza bruta se podría conseguir las claves de encriptado.
 - o Para evitar esta debilidad o intentar minimizarla:
 - Que el fichero donde se guardan las contraseñas encriptadas no tengan acceso todos los usuarios del sistema.
 - Contraseñas complejas.
 - Forzando que cada cierto tiempo se cambien las contraseñas.
 - ➔ A traves del comando **chage** fuerzo al usuario a cambiar la contraseña cada cierto tiempo. Modifica el contenido de /etc/shadow.
 - ➔ Para poner un determinado dia, la fecha no se pone en formato fecha, sino que se pone en dias desde el 1 de Enero de 1970 hasta el dia de la fecha que se requiera.
- El fichero **/etc/shadow**:

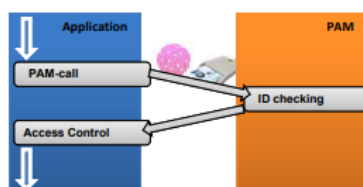
```
test:$1$decPVavi$ttM6DkmowBulunrnflgK90:11547:0:99999:7:::
```

- **Name:** username (same as passwd file).
- **Password:** encrypted password (\$algorithm\$salt\$hashed).
- **Date:** last change date (days since January 1, 1970)
- **Minimum:** number of days before next passwd change (if set to zero, no limits between changes).
- **Maximum:** upper limit for password modification.
- **Warning:** number of warning days previous to password caducity.
- **Inactive:** inactivity days after caducity(then locking)
- **Expire:** Account expiration date.

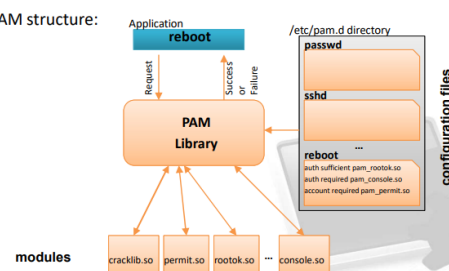


Unificación de Mecanismos de Autenticación:

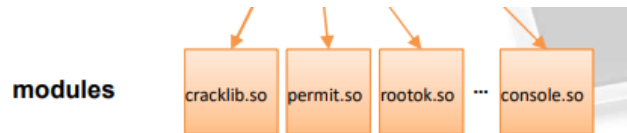
- Se delegan las tareas de autentificacion a un modulo externo **PAM**, ya que si se añade un nuevo metodo como huella dactilar, reconocimiento facial, etc. cambiar todo el software no es eficiente ni factible.
- **PAM: Pluggable Authentication Module:**
 - o Framework de autentificación, cuyas librerías permiten a los desarrolladores a abstraerse de los procesos de autentificación.
 - o Facilita la implementación de diferentes métodos y políticas de autentificación sin tener que modificar el código fuente de las aplicaciones.



Internal PAM structure:



- Existe un fichero de configuración por cada servicio/aplicación que haga un uso de PAM, en el directorio **/etc/pam.d**.
 - Determina qué módulos se utilizan y en qué orden se ejecutan para cada servicio de autenticación. Esto permite una gran flexibilidad y adaptabilidad, ya que los administradores del sistema pueden configurar PAM según sus necesidades y políticas de seguridad.
 - Los **módulos de PAM** son componentes independientes que realizan tareas específicas dentro del proceso de autenticación.
 - Se organizan en una pila y se ejecutan secuencialmente en función de la configuración. Cada módulo devuelve un resultado que determina el flujo del proceso de autenticación, y la configuración de PAM define qué módulos se utilizan y en qué orden para cada servicio de autenticación.



- **Módulo cracklib:** Su propósito es verificar la fortaleza de las contraseñas, con el fin de evitar que un usuario introduzca una contraseña demasiado simple y fácil.
 - La fortaleza se checkea de la siguiente forma:
 1. Comparación con diccionario: El módulo CrackLib puede comparar la contraseña con un diccionario de palabras comunes (por defecto en inglés). Si la contraseña coincide con una palabra del diccionario, se considera débil.
 2. Comparación con la contraseña anterior: El módulo puede verificar si la nueva contraseña es similar o demasiado similar a la contraseña anterior del usuario. Puede analizar si hay cambios suficientes, como cambios en mayúsculas/minúsculas o en caracteres diferentes, para determinar la fortaleza.
 3. Longitud y fortaleza: El módulo también evalúa la longitud de la contraseña para asegurarse de que cumpla con un mínimo establecido. Además, verifica si la contraseña contiene una combinación de diferentes tipos de caracteres, como letras mayúsculas y minúsculas, números y caracteres especiales.
 - Estos criterios pueden ser configurados y personalizados en el archivo de configuración del módulo CrackLib (**/etc/pam.d/common-password**). Allí se pueden establecer políticas específicas para determinar qué se considera una contraseña segura y qué se considera débil.
 - Al utilizar el módulo CrackLib, se pueden definir requisitos como una longitud mínima de contraseña, la exclusión de palabras comunes, la necesidad de una combinación de caracteres, entre otros. Si la contraseña no cumple con estos criterios, se considera débil y el usuario puede ser solicitado a proporcionar una contraseña más segura.

RESUMEN MODULO CRACKLIB: realiza el chequeo de la fortaleza de una contraseña mediante la comparación con diccionarios, la verificación de cambios respecto a contraseñas anteriores y la evaluación de la longitud y la composición de la contraseña. Estos criterios pueden ser personalizados según las políticas de seguridad establecidas en el archivo de configuración del módulo.

Delegación de Privilegios:

- A veces es útil habilitar ciertos permisos a usuarios que no son root.
 - o Se puede realizar mediante el comando **sudo**.
 - o Para habilitar ciertos privilegios, se deben declarar en el fichero **/etc/sudoers**

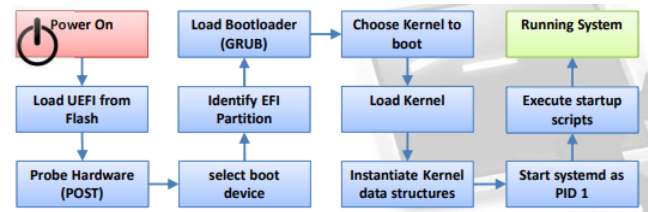
- **Example:** `%admin ALL=(ALL:ALL)NOPASSWD /usr/bin/apt-get`

- %admin: all the users in the admin group (single user without %)
- ALL: from any HOST/IP direction
- (ALL:ALL): can execute command as any user:group (not only root)
- NOPASSWD: no password required
- /usr/bin/apt-get: list, comm separated, of allowed commands.

* Ejemplo: Todos los usuarios en el grupo "admin" tendrán acceso a este comando.

Tema 4: Booting & Shutting Down

- Booting, Stage 1: **Hardware+UEFI**
- Booting, Stage 2: **Bootloader (GRUB)**
- Booting, Stage 3: **Kernel**
- Booting, Stage 4: **Systemd**



→ Antes el proceso el arranque y apagado se realizaba con la BIOS (y MBR) y SysV.

→ Ahora el proceso de arranque y apagado se realiza mediante UEFI (y GPT) y Systemd.

- El principal objetivo del arranque es cargar el kernel en memoria y empezar su ejecución.

Stage 1: Hardware:

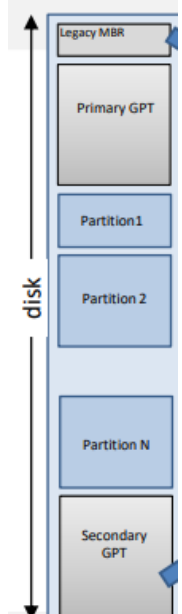
En primer lugar se pulsa el botón de encendido xd.

1. Inicialización del Firmware:

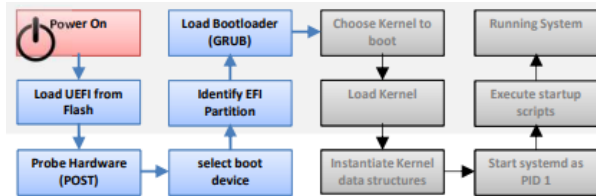
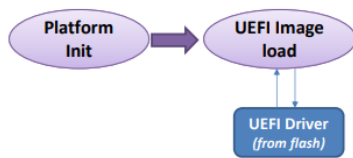
- Antes era la **BIOS: Basic Input Output System**
 - o Desarrollado por IBM en los 70s.
 - o Soporte para el estándar de MBR.
 - o Sistema de 16 bits.
 - o Sin driver de disco disponible → No se tiene noción de los ficheros/directorios del disco.
- Hoy en día **UEFI: Unified Extensible Firmware Interface**
 - o Desarrollado por Intel en los 2000.
 - o Mejor soporte de disco/red: soporte completo para la Tabla de Particiones GUID (GPT) y IPv6.
 - o Controlador para "entender" el sistema de archivos FAT (formato de la Partición del Sistema EFI o ESP).
 - o No hay una necesidad real de un cargador de arranque (no se requiere GRUB).
 - o Mejora de la seguridad. **UEFI Secureboot** (ejecución solo de aplicaciones firmadas) para evitar malware previo al sistema operativo (bootkit).
 - o Integración más estrecha entre el sistema operativo y el entorno de prearranque: requiere soporte del sistema operativo (Linux, OSX, Windows 10).

Recordamos: GPT Disk & Partitions

- **MBR**: Sirve para mantener cierta compatibilidad con las estructuras de particionado de disco antiguas.
 - o Se hace una definicion de un disco como si fuese MBR, de esta forma los procesos que se crean que están trabajando con un disco MBR puedan seguir con sus tareas.
- **Secondary GPT Header**: Back up por si nuestro gpt primario se rompe.



Stage 1: Hardware + UEFI (Boot Manager):

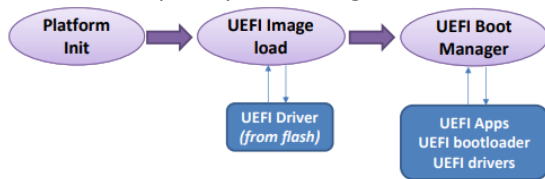


1º)

- Inicializa la memoria principal, así como los componentes de hardware requeridos para las siguientes fases.
- Carga los controladores de dispositivo desde la memoria flash a la memoria principal (ahora disponible), inicializa todo el hardware requerido (disco (sistema de archivos FAT), monitor, teclado, ...) y registra/utiliza protocolos.
 - o Protocolo: proporciona salida de texto a la consola o acceso a un dispositivo PCI.

2º)

Una vez la máquina ya está cargada:



- **UEFI Boot Manager:** intenta cargar aplicaciones UEFI (.efi files) en un orden predefinido. → efibootmgr -v
 - o Las aplicaciones pueden ser: cargadores de sistema operativo, núcleos EFI, controladores adicionales (ext4), shell, GUI, etc.
 - o Las variables de NVRAM (memoria no volátil) definen este orden
- Las aplicaciones deben residir en un sistema de archivos definido por UEFI:
 - o Sistema de archivos FAT (formato de la Partición del Sistema EFI o ESP).
- ¿Cómo se realiza el proceso de búsqueda de aplicaciones y cargadores?
 - o UEFI consulta la tabla de particiones GPT para identificar la **ESP(Partición VFAT)**.
 - Lee el archivo de aplicación objetivo (.efi) desde un archivo en la ESP y lo ejecuta.
 - Ruta del archivo a cargar: parámetro de configuración. Por defecto (Debian):
/efi/boot/bootx64.efi
 - ➔ Cada sistema operativo instalado tiene su propio directorio en la partición EFI.
 - o **Si no se utiliza un cargador de arranque** (soporte EFI stub habilitado en el kernel), todos los archivos necesarios para cargar el sistema operativo (kernel, ramdisk, etc.) deben estar disponibles en esta partición.

RESUMEN UEFI: UEFI inicializa la memoria principal y los componentes hardware, carga los controladores de la memoria flash a la memoria principal y finalmente carga el fichero .efi de la partición de arranque ESP(VFAT), en cuyo fichero está alojado el bootloader(GRUB) además de todos los controladores necesarios para inicializar el sistema hardware.

- Se puede elegir el modo de acceso mediante "**UEFI graphical interface**":
 - o La ejecución de la máquina se carga mediante el Boot Manager, en el cual hay presentes distintas formas de "inicio de sesión":
 - **UEFI Shell:**
 - ➔ Hay que tener en cuenta que EFI es un firmware (pequeño SO instalado en la placa base) y por lo tanto puede tener su propia Shell.
 - **UEFI Graphical User Interface:**
 - ➔ En muchos casos en vez de la shell anterior nos encontraremos esta interfaz, aun así el objetivo de ambos es el mismo, interactuar con el sistema.

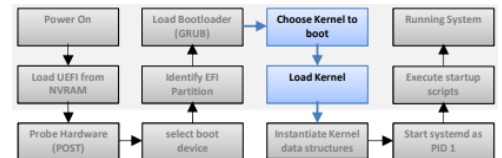


Configuración del UEFI Boot Manager:

- Tenemos 3 formas para configurarlo:
 1. A través de la **UEFI Shell**:
→ Comando bcfg
 2. A través de la **UEFI Graphical Interface**
 3. **Una vez ya se haya arrancado el sistema**:
→ Comando efibootmgr

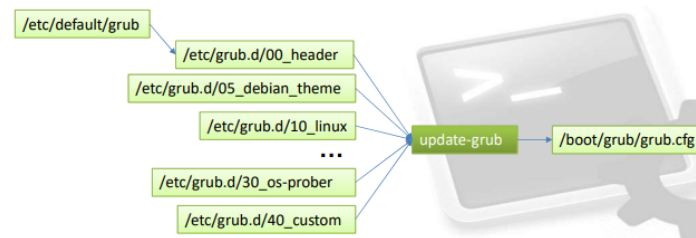
Stage 2: Bootloader (GRUB):

- Esta etapa intermedia se mete por 2 razones:
 - o Para proporcionarnos un sistema multiarranque.
 - o Para configurar como quiero que arranque el kernel.
- Su tarea es cargar el kernel en memoria principal y que empiece a funcionar.
- **GRUB: Grand Unified Bootloader**
 - o Su actual utilidad es proporcionar un sistema de multiarranque.



Configuración del GRUB:

- **GRUB** lee el **fichero /boot/grub/grub.cfg**, es un fichero de texto que normalmente no se edita a mano, sino que se editan a mano los ficheros que se usan para editar el grub.cfg, presentes en el **directorio /etc/grub.d** y **fichero /etc/default/grub**.

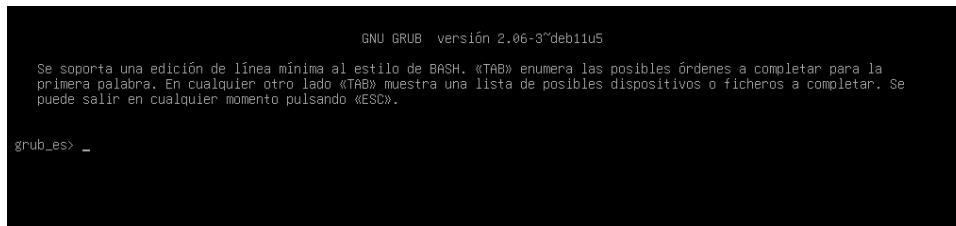


- Fichero **/etc/default/grub**:
 - o Este archivo contiene variables y opciones de configuración que afectan al comportamiento y apariencia de GRUB durante el proceso de arranque.
 - o Aquí hay algunas líneas comunes que se pueden encontrar en el archivo "/etc/default/grub" y su explicación:
 1. **GRUB_DEFAULT**: Esta variable determina la entrada de arranque predeterminada que se seleccionará en GRUB. Puede especificarse como un número (índice de entrada) o como el nombre de la entrada.
 2. **GRUB_TIMEOUT**: Especifica el tiempo de espera en segundos que GRUB muestra el menú de arranque antes de iniciar automáticamente la entrada predeterminada.
 3. **GRUB_CMDLINE_LINUX**: Esta variable permite agregar opciones de línea de comandos al kernel Linux durante el arranque. Aquí se pueden agregar opciones como parámetros de configuración del sistema, opciones de controladores, etc.
 4. **GRUB_DISABLE_RECOVERY**: Si se establece en "true", oculta las entradas de recuperación en el menú de arranque de GRUB.
 5. **GRUB_THEME**: Permite especificar una ruta al archivo de tema de GRUB que se utilizará para personalizar la apariencia del menú de arranque.
 - Estas son solo algunas de las variables y opciones comunes que se pueden encontrar en el archivo "/etc/default/grub". Es importante tener cuidado al modificar este archivo, ya que los cambios incorrectos pueden afectar la capacidad de arranque del sistema.

- Directorio **/etc/grub.d**:
 - Contiene una serie de scripts de configuración que se utilizan para generar/modificar el archivo de configuración principal de GRUB, llamado "**grub.cfg**".
 - Cada script en el directorio **"/etc/grub.d"** es responsable de configurar una parte específica del menú de arranque de GRUB.
 - Por ejemplo, puede haber scripts para configurar las entradas de arranque para diferentes sistemas operativos instalados, scripts para configurar opciones de arranque avanzadas, scripts para configurar opciones de recuperación, entre otros. Algunos son:
 - ➔ **00_header**: Este script es responsable de generar la cabecera del archivo "grub.cfg" y realizar configuraciones generales, como la configuración de variables y ajustes iniciales.
 - ➔ **10_linux**: Este script se encarga de generar las entradas de arranque para los diferentes kernels de Linux instalados en el sistema.
 - ➔ **30_os-prober**: Este script busca otros sistemas operativos instalados en el disco y genera las entradas de arranque correspondientes. Por ejemplo, si tienes un sistema operativo Windows instalado en otra partición del disco, este script generará una entrada de arranque para Windows en el menú de GRUB.
 - ➔ **40_custom**: Este script permite agregar entradas de arranque personalizadas. Puedes editar este script para agregar tus propias entradas de arranque o configuraciones especiales.

IMPORTANTE: Después de realizar cambios en estos scripts o en el fichero **/etc/default/grub**, se debe ejecutar el comando **"update-grub"** para generar o actualizar el archivo "grub.cfg" en función de los scripts y/o variables modificadas.

Grub Command Line:



```
GNU GRUB versión 2.06-3~deb11u5

Se soporta una edición de línea mínima al estilo de BASH. «TAB» enumera las posibles órdenes a completar para la
primera palabra. En cualquier otro lado «TAB» muestra una lista de posibles dispositivos o ficheros a completar. Se
puede salir en cualquier momento pulsando «ESC».

grub_es> _
```

- Interfaz de texto que proporciona un entorno interactivo para ejecutar comandos y realizar tareas relacionadas con el gestor de arranque GRUB.
 - Permite editar la configuración de arranque del kernel durante el proceso de arranque.
 - También puedes arrancar un sistema operativo no listado.
 - Mostrar información del sistema.
 - Realizar pruebas de sistema de archivos.

➔ **Hay que proteger las etapas 1 y 2 del arranque (Hardware + UEFI // Bootloader GRUB)**, ya que si se tiene acceso físico al ordenador podría suponer una gran debilidad a la seguridad del sistema:

- Proteger las modificaciones del boot, ya que podrían obtener acceso a privilegios del root.
 - Desde la grub Shell se podría modificar el contenido del fichero grub.cfg, permitiendo modificar las características del arranque del sistema y habilitando el acceso al atacante.
 - Se llegaron a detectar hasta backdoors en el [secureboot](#) implementado por UEFI debido a overflows en el heap de los scripts de grub.d. (Ya solucionado)

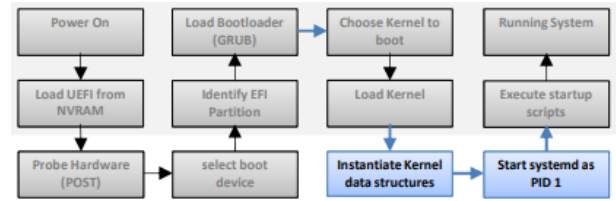
Stage 3: Kernel:

Una vez el bootloader ha cargado el kernel y los ficheros ramdisk en memoria:

→ vmlinuz-4.9...

→ initrd-4.9...

Loading the kernel:



1. El kernel se descomprime a sí mismo.
2. Detecta el mapa de memoria, la CPU y las características que admite.
3. Arranca la consola para mostrar por pantalla información.
4. Checkea el bus PCI, creando una tabla con los periféricos detectados.
5. Inicializa el sistema encargado de la gestión de la memoria virtual, incluido el swap.
6. Inicializa los drivers de los periféricos detectados (monolíticos o modulares).
7. Monta el sistema de ficheros del directorio root “/”.
8. Llama a los procesos de systemd (Stage 4), PID 1 como padre del resto de procesos.

The init ramdisk:

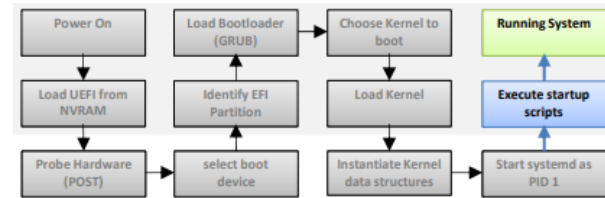
- **Ramdisk:** Fracción de memoria principal (RAM) formateada con un sistema de ficheros (tmpfs, ramfs).
 - o Actúa como sistema de ficheros.
 - o Es volátil, una vez se desmonte se perderán todos sus datos.
 - o Es necesario dado a que el kernel suele ser modulado, quiere decir que parte de sus funcionalidades están en sda y otras partes en la memoria principal.
- **Init Ram Disk (Initrd):** Fracción de memoria principal (RAM) usada para agilizar el arranque del sistema con el fin de obtener los elementos necesarios para cargar el kernel en el directorio root.
 - o Se realiza un montaje previo del initrd que del directorio root.
 - o Es un sistema de ficheros temporal, para que cuando se ejecuta el kernel posteriormente pueda tener acceso al directorio root “/”.
 - Cuando el sistema arranca, el bootloader (como GRUB) carga el initrd en la memoria RAM y lo pasa al kernel durante el proceso de arranque.
 - El kernel luego descomprime y monta el initrd como un sistema de ficheros temporal.
 - A partir de ahí, el initrd se utiliza para cargar los controladores de dispositivos necesarios para reconocer y montar el sistema de archivos raíz real.
 - Una vez que el sistema de archivos raíz está montado, el initrd se desmonta y se libera de la memoria RAM.

→ Su principal objetivo es cargar los módulos necesarios para realizar el montaje del sistema de ficheros del directorio root “/”.

RESUMEN INITRD: Imagen de sistema de ficheros temporal utilizada durante el inicio del sistema operativo Linux. Proporciona los controladores de dispositivos y módulos del kernel necesarios para arrancar el sistema y montar el sistema de archivos raíz real. Una vez que el sistema de archivos raíz está montado, el initrd se desmonta y se libera de la memoria RAM.

Stage 4: Systemd:

- Una vez cargado el kernel, se ejecuta el daemon de administración del sistema (**/sbin/init**).
- El objetivo principal es asegurarse de que el sistema ejecute el grupo correcto de servicios (modo) y daemons en cualquier momento dado.
- Proporciona un administrador de sistema y servicios que se ejecuta como PID 1.
- Se encarga de tareas de inicio como: configurar el nombre de la computadora y la zona horaria, verificar el estado del disco, montar sistemas de archivos, configurar interfaces de red, etc.
- Implementaciones alternativas: SysV-init, BSD-init, systemd.
- ¿Cuáles son las **principales características de systemd**?
 - o Capacidad de paralelización agresiva (arranque más rápido).
 - o Activación de sockets y D-Bus para iniciar servicios, inicio bajo demanda de daemons.
 - o Realiza un seguimiento de los procesos utilizando grupos de control de Linux.
 - o Mantiene los puntos de montaje y montaje automático.
 - o Daemon de registro de eventos.
 - o Utilidades para controlar la configuración básica del sistema.
 - Nombre de host, fecha, configuración regional, lista de usuarios conectados, cuentas de sistema, directorios y configuraciones en tiempo de ejecución.
- Mientras que **SysV-init** se centra principalmente en el proceso de arranque del sistema, **systemd** va más allá y proporciona una solución integral para la gestión del sistema desde el arranque hasta la ejecución de procesos y servicios. ([Más explicación](#)).



SysV-init is a booting management daemon.
Systemd is a booting and run-time management daemon.

→ **Systemd** es un daemon de administración y servicio que se utiliza en sistemas operativos Linux. Proporciona un conjunto de herramientas y servicios para el inicio, el control y la supervisión del sistema.

→ Un **daemon** es un programa o proceso informático que se ejecuta en segundo plano de forma continua, sin interacción directa con los usuarios del sistema.

- **Systemd Unit**: Archivo de configuración que define cómo se debe iniciar, detener, reiniciar y administrar un servicio o recurso en el sistema operativo Linux.
 - o Las unidades systemd pueden ser de diferentes tipos, y cada tipo tiene su propósito y comportamiento específicos.
 - o Por ejemplo, una **unidad .service** representa un servicio del sistema, como un servidor web o un servicio de base de datos. Una **unidad .target** agrupa otras unidades y define un grupo de unidades que deben iniciarse juntas para lograr un objetivo específico.
 - (Estas dos unidades están envueltas en el proceso de arranque).
 - o Otras unidades como .automount, .device, .mount, .socket, .swap, .timer, entre otras, representan diferentes tipos de recursos o configuraciones que pueden ser administrados por systemd.
- Systemd Unit Syntax:
 - o Los ficheros unit comparten la sintaxis.
 - o Estructura organizada por secciones.
 - Cada sección se denota utilizando corchetes y tiene un nombre específico, como [Service], [Timer], [Mount], etc. Estas secciones definen diferentes aspectos y comportamientos de la unidad.
 - Dentro de cada sección, se utilizan directivas de clave-valor para configurar y definir el comportamiento de la unidad. Cada directiva se especifica en una línea separada y sigue el formato de "clave = valor". La clave representa una propiedad o configuración específica de la unidad, mientras que el valor define el contenido o el ajuste para esa propiedad.

Directivas clave-valor

```
[Unit]
Section
Description=fast remote file copy program daemon
ConditionPathExists=/etc/rsyncd.conf

[Service]
ExecStart=/usr/bin/rsync --daemon --no-detach

[Install]
WantedBy=multi-user.target
```

- Localización de los Unit Files:
 - `/lib/systemd/system/` y/o `/etc/systemd/system/`
 - Cuando haya algún proceso que comparta unit files en ambos directorios, prevalece la dirección del `/etc`.
 - Se pueden encontrar, en `debian`, en ambos directorios.
- Directivas de la sección [Unit]: Define metadatos de la unidad y la relación con otras unidades.
 - La directiva Description/Documentation se utiliza para describir el nombre de la unidad y proporcionar información básica sobre su funcionalidad. También puede indicar la ubicación de la documentación relacionada con la unidad.
 - Las directivas Requires y Wants se utilizan para establecer las dependencias de la unidad actual. Requires establece dependencias estrictas, lo que significa que las unidades enumeradas deben activarse si se activa la unidad actual. Por otro lado, Wants establece dependencias más flexibles, donde las unidades enumeradas no necesariamente deben activarse si se activa la unidad actual.
 - Existen otras directivas como Requisite, Binds To, PartOf y Conflicts que ofrecen formas adicionales de gestionar las dependencias y las interacciones entre unidades.
 - Las directivas Before y After se utilizan para controlar el orden de inicio de las unidades. Before indica que las unidades enumeradas no se iniciarán hasta que la unidad actual se marque como iniciada, mientras que After garantiza que las unidades enumeradas se iniciarán antes de iniciar la unidad actual.

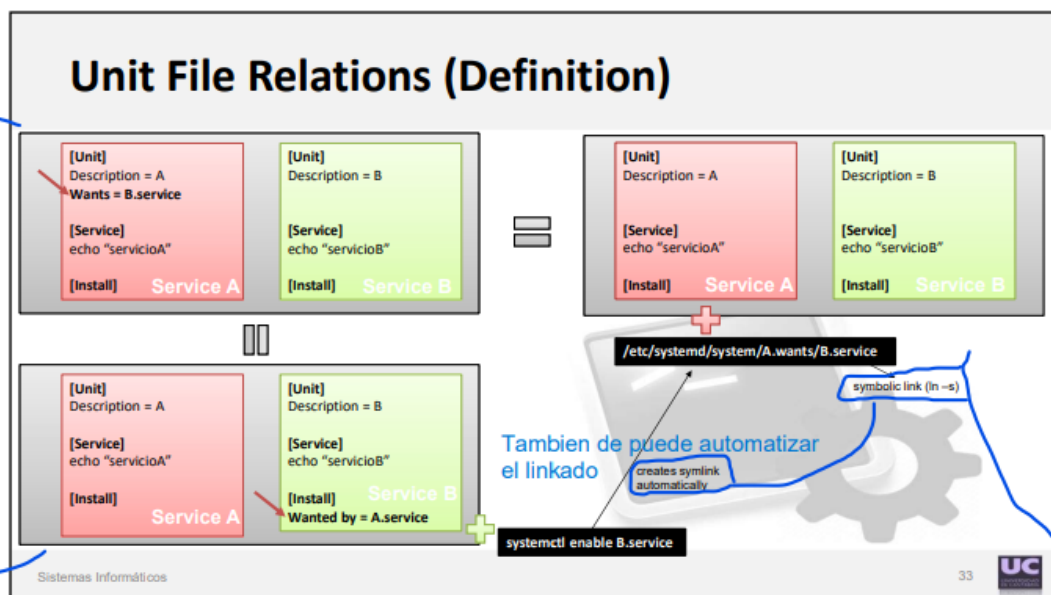
→ En resumen, las directivas de la sección [Unit] en los archivos de unidad de `systemd` proporcionan información sobre la unidad, establecen dependencias con otras unidades y controlan el orden de inicio de las unidades. Esto permite una gestión eficiente y controlada de las interacciones y dependencias entre las unidades administradas por `systemd`.
- Directivas de la sección [Install]: Define el comportamiento de una unidad cuando se habilita o deshabilita (`systemctl enable/disable`).
 - La directiva WantedBy se utiliza para especificar una dependencia similar a la directiva Wants en la sección [Unit]. Cuando una unidad con la directiva WantedBy se habilita, se crea un directorio en la ubicación `/etc/systemd/system/[unit].wants`. Dentro de este directorio se crea un enlace simbólico que establece la dependencia entre las unidades.
 - Indica qué unidades desean tener activa la unidad actual.
 - La directiva RequiredBy se utiliza para especificar las unidades que requieren la unidad actual. Si la unidad actual se habilita, las unidades enumeradas en la directiva RequiredBy también se habilitarán automáticamente.
 - Establece una dependencia fuerte en la que una unidad es necesaria para el funcionamiento de otra unidad

→ Estas directivas en la sección [Install] permiten definir cómo se comporta una unidad cuando se habilita o deshabilita en el sistema. Proporcionan un mecanismo para establecer dependencias entre unidades y asegurarse de que las unidades necesarias se activen o desactiven adecuadamente. Esto es útil para controlar el inicio y apagado de servicios y otros recursos administrados por `systemd`.
- Directivas de la sección [Service]: Realizan la configuración del servicio.
 - Type: categoriza el servicio según su proceso y comportamiento de demonización.
 - `simple`: tipo por defecto.
 - `forking`: el servicio crea un proceso hijo.
 - `oneshot`: `systemd` espera a que el proceso finalice antes de continuar con otras unidades.
 - `dbus`: la unidad tomará un nombre en el bus de D-Bus.
 - `notify`: el servicio emitirá una notificación cuando haya terminado de iniciar.
 - `idle`: el servicio no se ejecutará hasta que se despachen todos los trabajos.
 - ExecStart: especifica la ruta y los argumentos del comando que se ejecutará para iniciar el servicio.

- ExecStop: especifica el comando necesario para detener el servicio.
- Restart: define las condiciones para intentar reiniciar automáticamente el servicio (siempre, en caso de éxito, en caso de error, ...).
- TimeoutSec: especifica el tiempo de espera antes de marcar el servicio como fallido (o forzar su finalización) al detenerlo.

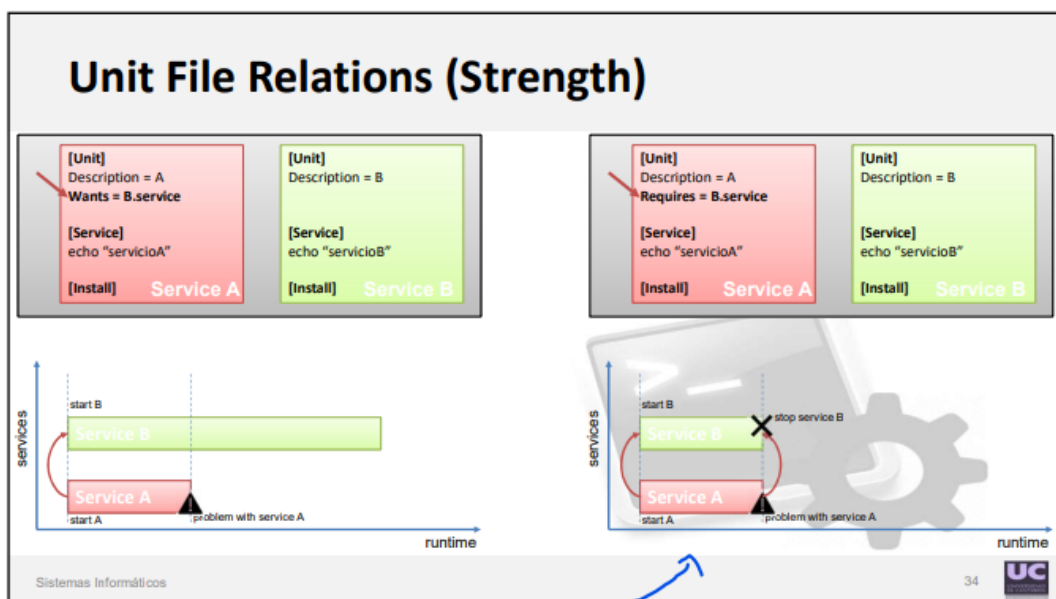
→ Estas directivas permiten configurar cómo se inicia, se detiene y se administra el servicio en el entorno de systemd. Proporcionan flexibilidad y control sobre el comportamiento del servicio durante su ciclo de vida.

Si el servicio A necesita al servicio B, tienen que en su sección unit meter una directiva "Wants = B.service" o también desde el Servicio B defino la dependencia hacia el servicio A con "Wanted by = A.service"



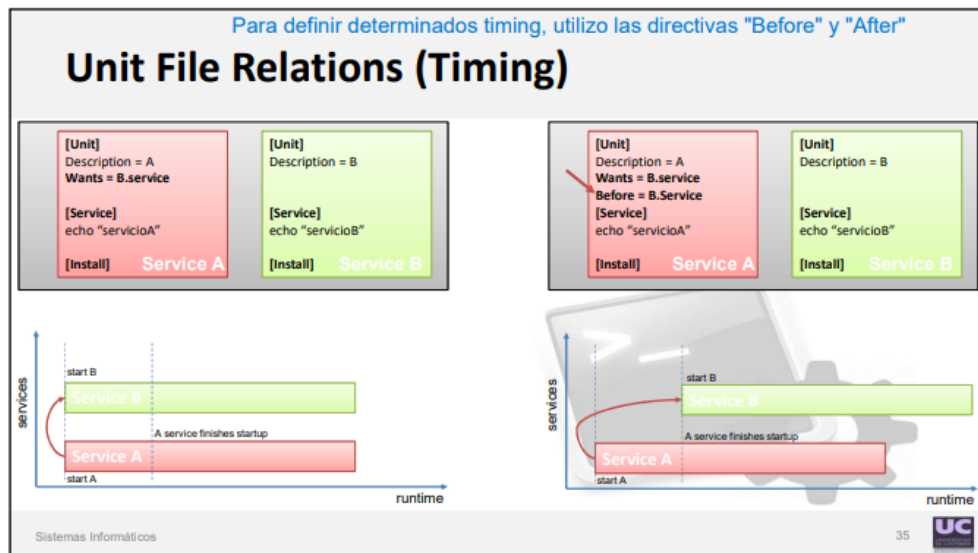
33

Equivalente a que en el fichero `/etc/systemd/system` creamos un directorio que se llame `A.wants` y otro dentro de `A.wants` meto un enlace simbólico que apunta a la dependencia llamada `B.service`



34

Si queremos que cuando por ejemplo falle el servicio A, todos los demás servicios asociados a el paren, debo plasmarlo mediante el `"Requires = B.service"`



- Una unit de tipo .target solo va a tener definidas directivas de tipo "Wants" y "Requires".

Systemd Boot Process:

- El arranque y manejo de los servicios se realiza mediante **Targets**.
 - o Unidades especiales utilizadas para agrupar las unidades de arranque y procesos de sincronización iniciales.

```
[Unit]
Description=foo boot target
Requires=multi-user.target
Wants=foobar.service
After=rescue.service
```

Como crear un servicio local en nuestro sistema:

1. Creamos nuestro Unit File.
2. Manejamos las dependencias convenientes/necesarias.
3. Guardamos el unit file en el directorio **/etc/systemd/system**.
4. Activamos las dependencias. → `systemctl enable [servicioDependiente].service`

Apagado:

- Un mal apagado puede suponer una pérdida en la consistencia y de datos en nuestro sistema. (Tema 2)
- Pasos para un correcto apagado:
 1. Advertir a todos los usuarios previamente.
 2. Detener todos los servicios asociados al objetivo.
 3. Enviar la señal específica a todos los procesos para finalizar su ejecución.
 4. Finalizar usuarios y procesos que aún estén presentes.
 5. Apagar los subsistemas de forma secuencial.
 6. Desmontar el sistema de archivos (sincronizando cambios pendientes con el disco).

Configuraciones Antiguas:

Firmware BIOS:

MBR & Disk Partitions:

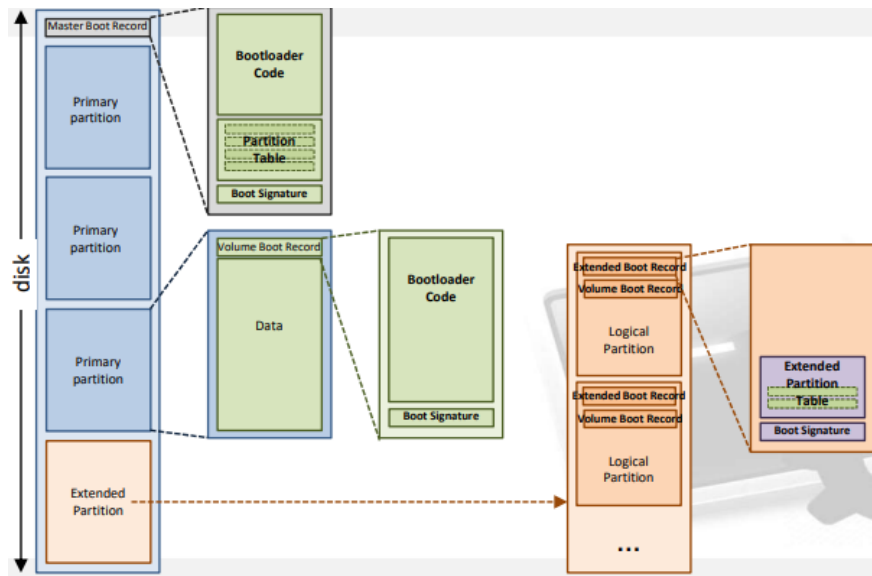
- o Primer bloque del disco, 512 bytes.
- o Tabla de particiones: contiene información sobre las particiones primarias en un disco. Cada entrada de la tabla de particiones describe una partición específica e incluye detalles como el inicio y el tamaño de la partición, el tipo de sistema de archivos utilizado y otros atributos relacionados

Volume Boot Record (VBR):

- Primer bloque de cada partición primaria.
- Puede contener código del gestor de arranque (indicado por la firma de arranque).

Extended Partition:

- Una partición que puede subdividirse en múltiples particiones lógicas.
- Registro de arranque extendido (EBR): primer bloque de cada partición lógica. Solo contiene una tabla de particiones con dos campos. La tabla de particiones extendidas forma una lista enlazada con todas las particiones lógicas.



Linux Naming Convention

Bootloader in MBR:

El bootloader en el contexto del MBR (Master Boot Record) funciona de la siguiente manera:

1. Durante el proceso de arranque, el firmware de la computadora (BIOS o UEFI) carga el MBR, que es el primer sector del disco.
2. El MBR contiene un pequeño código de bootloader, generalmente conocido como código de arranque, que se encuentra en los primeros 446 bytes del sector MBR.
3. El firmware ejecuta este código de arranque del MBR, que tiene la tarea de buscar la partición activa o la partición de arranque.
4. Una vez que se encuentra la partición activa, el código de arranque carga el primer sector de la partición, que se conoce como Registro de Arranque del Volumen (VBR, Volume Boot Record).
5. El VBR contiene el código del bootloader específico del sistema operativo instalado en esa partición. Este código del bootloader del VBR se carga en la memoria y se ejecuta.
6. El bootloader del VBR tiene la responsabilidad de cargar y transferir el control al kernel del sistema operativo correspondiente.
7. A partir de ese momento, el kernel del sistema operativo se hace cargo del proceso de arranque y continúa cargando el sistema operativo completo.

En resumen, el bootloader en el MBR actúa como un intermediario entre el firmware de la computadora y el sistema operativo instalado, cargando el código del bootloader específico del sistema operativo y transfiriendo el control para iniciar el sistema operativo correctamente.

LILO (Linux Loader):

- Bootloader de dos etapas.
- No entiende de sistema operativo ni de sistema de ficheros.

→ Pasos de arranque con LILO:

1. El master boot carga LILO de la primera partición activa y lo ejecuta.
 - a. Puede estar ubicado en el MBR o en el boot block de una partición primaria. En el segundo caso, MBR contiene el código necesario para cargar LILO en otro bloque.
 2. LILO pregunta al usuario que tipo de arranque precisa(partición, kernel, modo...).
 - a. Mediante un prompt.
 3. LILO carga el kernel y el ramdisk.
 4. El kernel empieza a funcionar una vez esté cargado en memoria.
- La configuración de LILO se realiza mediante el **fichero /etc/lilo.conf**.
 - o Cada cambio realizado en el proceso de arranque debe ser reflejado en la configuración de LILO.
 - Un error en el arranque no puede ser arreglado desde la Shell.
 - Posibles errores:
 - o Instalación de un SO que sobrescriba el MBR.
 - o La compilación del kernel fallida.
 - o Modificación de los boot files sin actualizar la configuración de LILO.
 - Sistemas de recuperación de fallos:
 - o Mkbootdisk.
 - o CD de recuperación.

INIT (SysV):

- El proceso de INIT realiza los siguientes pasos:
 1. PASO 1 → Configuración: Lee del **fichero /etc/inittab** la configuración inicial del sistema (runlevels, consolas...).
 2. PASO 2 → Inicialización: Ejecuta el **comando /etc/init.d/rc.S** (debian), que realiza la inicialización básica del sistema.
 3. PASO 3 → Servicios: De acuerdo con el [runlevel](#) configurado, ejecuta los scripts/servicios preestablecidos para ese runlevel.

El **runlevel**, también conocido como nivel de ejecución, es un concepto utilizado en los sistemas operativos Unix y Linux para definir los diferentes modos de funcionamiento del sistema. Representa un estado específico en el que se encuentra el sistema y determina qué servicios y procesos se ejecutan en ese momento.

En los sistemas basados en SysV (System V), se utilizan números del 0 al 6 para representar los diferentes runlevels. Cada runlevel tiene asociado un conjunto de servicios y procesos que se activan o desactivan según el nivel.

Runlevels (Operation modes)

- Standard: 7 levels. Each distribution its own configuration (here Debian)
- Level **S**: only executed at boot time (replaces /etc/rc.boot)
- Level **0**: **Halt**. Employed to Shut down the system.
- Level **1**: **Single User**. Maintenance tasks (no active network)
- Level **2-5**: **Multiuser**. All the network and Graphical services activated.
- Level **6**: **Reboot**: Similar to level 0.

En contraste con el enfoque secuencial de SysV, [systemd](#) utiliza un enfoque más orientado a la paralelización y la concurrencia durante el proceso de arranque y ejecución de servicios iniciales.

Systemd es un sistema init moderno y ampliamente utilizado en muchas distribuciones de Linux. Su objetivo es mejorar la velocidad de arranque y la eficiencia del sistema al permitir que los servicios se inicien de forma paralela en lugar de secuencial. En lugar de esperar a que un servicio se inicie completamente antes de pasar al siguiente, systemd puede iniciar varios servicios al mismo tiempo, aprovechando al máximo la capacidad de procesamiento de la máquina.

Esto se logra mediante el uso de dependencias entre los servicios y la implementación de unidades systemd. Las unidades pueden especificar las dependencias entre sí y systemd se encarga de gestionar el orden de inicio de los servicios para garantizar que se cumplan todas las dependencias.

Tema 5: Resource Management & Logging

¿Porqué nos debemos preocupar del rendimiento?:

Perspectiva financiera: mejorar la relación precio/rendimiento es un aspecto clave en entornos de nube.

Planificación futura: planificación de capacidad, eliminación de cuellos de botella, análisis de escalabilidad, etc.

Gestión multiusuario: evitar comportamientos abusivos de ciertos usuarios en relación con determinados recursos compartidos.

- El status del kernel está alojado en el **directorio /proc** (pseudo file system).
 - o Muestra información del hardware.
 - o Muestra información de los procesos en ejecución.
 - o No se suele leer directamente, existen comando para ello como **ps**, **vmstat**...
- Sirven para monitorizar y controlar los procesos del sistema, con el fin de:
 - o Detectar programas con bugs.
 - o Detectar comportamientos inadecuados.
 - o Detectar usuarios maliciosos o usuarios ignorantes de sus actos.
- Las acciones a tomar son:
 - o Prevención: limitando la cantidad de recursos a cada usuario.
 - o Monitorización y corrección: Revisiones periódicas y correcciones.

Comandos del tema 5:

- Monitorización de procesos:
 - o **ps**: Monitorización de procesos
 - o **htop**: Monitorización de procesos en tiempo real.
 - o **vmstat**: Utilización instantánea global.
 - o **uptime**: utilización media global.
 - o **strace**: Info adicional sobre un proceso.
 - o **free**: Información de disponibilidad de la memoria principal.
 - o **df**
 - o **du**
- Monitorización de procesos Logging:

Los procesos que se ejecutan en el sistema dejan una serie de mensajes informativos en el log, estos mensajes o información no es legible directamente ya que su contenido es binario, por lo que accederemos a ello mediante el comando journalctl.

 - o **Journalctl**

Estos mensajes son generados por kernel, servicios y apps, que mandan información normal, problemas obtenidos en el arranque o información con relación a la seguridad.
- Configuración del Logging:

Los cambios en la configuración del log se realizarán a través del **fichero /etc/systemd/journald.conf**:

 - Nivel de prioridad de mensajes que salen por pantalla.
 - Donde guardar el journal.
 - Limites de los ficheros journal. (Capacidad máxima, número máximo de ellos, periodo de vigencia...)
- Manejo de procesos Señales:
 - o **Kill**

Envia determinadas señales a procesos.

El comando kill no mata un proceso, sino que le manda una determinada señal con un fin definido.

Lo que mata un proceso es el comando kill con la opción -9 (señal de matado).
- Manejo de procesos Prioridad:

Si hay procesos que gastan mucha cpu, se le puede bajar la prioridad para que así la cpu se reparta mas entre todos los procesos.

 - o **nice**
 - o **renice**

- Limitación de recursos a nivel de Shell:

o **Ulimit**

El comando ulimit limita la utilización de recursos por parte del usuario en la Shell.

Para hacer estas limitaciones permanentes en el sistema, se declaran en el fichero: **/etc/security/limits.conf**

- Para configurar a los usuarios los recursos que le asignaras, como por ejemplo %cpu a alumno.
- Los tipos de limitaciones pueden ser **Hard** o **Soft** → El limite hard solo lo puede asignar el usuario root y el soft puede asignarlo cualquier usuario y quitarlo tambien.

```
calderon:~> ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
file size               (blocks, -f) unlimited
max locked memory       (kbytes, -l) unlimited
max memory size         (kbytes, -m) unlimited
open files              (-n) 256
pipe size               (512 bytes, -p) 1
stack size              (kbytes, -s) 8192
cpu time                (seconds, -t) unlimited
max user processes      (-u) 709
virtual memory          (kbytes, -v) unlimited
```

- Manejo de memoria **Swap**:

Es necesario ya que si la memoria que necesita un sistema para ejecutar determinados programas, es menor que la que se dispone, estos programas nunca se ejecutarán.

o **systemctl**

- Modificación del Swap:

Existen 2 alternativas para modificar el tamaño del swap:

1. Añadiendo una **partición nueva del tipo Swap**:
2. Crear un **swap file**:

Con systemd con ficheros.

o **systemctl**

```
[Unit]
Description= Turn on swap

[Swap]
What=/swapfile

[Install]
WantedBy=multi-user.target
```

- Gestión de discos:

Limitar la capacidad por usuario, asignandoles particiones o limites de memoria usables.

- o **edquota**
- o **quotaon**
- o **quotaoff**
- o **repquota**
- o **quota -v user**

- Ejecución de Procesos Periódicamente:

Delegar en el tiempo una tarea, es decir, que una tarea se ejecute en un momento determinado.

Se puede gestionar a traves de 2 formas:

1. **Fichero crontab**: /etc/crontab

Realizando una nueva entrada del comando/acción a automatizar en el fichero /etc/crontab.



2. **Systemd timer**:

Creo un servicio para la acción a realizar y posteriormente un timer asociado al servicio.

```
[Unit]
Description= A timer that runs the service the
first four days of each month at 12:00 PM,
but only if that day is a Monday or a Tuesday

[Timer]
OnCalendar= Mon,Tue *-*-01..04 12:00:00
Persistent=true

[Install]
WantedBy=timers.target
```

```
[Unit]
Description= A timer that starts 15 minutes after
boot and again every week while the system is
running

[Timer]
OnBootSec=15min
OnUnitActiveSec=1w

[Install]
WantedBy=timers.target
```

Por cada fichero .timer tiene que existir otro fichero con el mismo nombre.

- Ejecución de proceso en un tiempo determinado:

- **Tab**

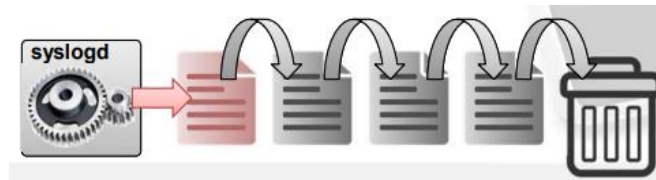
- Configuración **Rsyslog**:

Rsyslog es un demonio de registro de eventos y mensajes que se utiliza para recopilar, procesar y almacenar registros de manera centralizada. Permite recopilar registros de múltiples sistemas y aplicaciones, proporciona capacidades de filtrado y enrutamiento avanzados, y admite varios protocolos de red para el envío de registros.

- Su configuración se realiza mediante el **fichero /etc/rsyslogd.conf**.
 - Existen tres partes:
 - Módulos.
 - Directivas.
 - Reglas.

- **Rotación del Log:**

La rotación de registros (log rotation) es un proceso mediante el cual los archivos de registro (logs) se administran y se limita su tamaño para evitar que crezcan indefinidamente y consuman demasiado espacio en disco.



Se puede gestionar de 2 formas:

1. **Manualmente:** implementando un **script** que lo realice.

```
#!/bin/sh
cd /var/log/
mv messages.2 messages.3
mv messages.1 messages.2
mv messages messages.1
cat /dev/null > messages
chmod 600 messages
#Reiniciar syslog
service restart rsyslog
```

2. **Ficheros logrotate.**

```
# rotate log files weekly, monthly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# send errors to root
errors root
# create new(empty)log files after rotating old ones
create
# compressed log files
compress
# DEB packages drop log rotation info into this dir
include /etc/logrotate.d
#no packages own lastlog or wtmp, rotate them here
/var/log/wtmp cd /var/log/{
    monthly
    create 0664 root utmp
    rotate 1
}
```

```
/var/log/dpkg.log {
    monthly
    rotate 12
    compress
    notifempty
    create 0664 root adm
}
```

Tema 6: Software Management

Instalación desde el Código Fuente:

Ventajas:

- Optimizar el software para nuestro hardware (opciones de compilación).
- Tener más libertad en cuanto a las versiones o software que se instalan.

Desventajas:

- Proceso de instalación complicado (compilación y dependencias).
- Es más fácil desorganizar la instalación del sistema.
 - > El software instalado no está etiquetado en ninguna base de datos.
 - > Se recomienda utilizar directorios especiales, como /usr/local/, /opt/ y /usr/src/.

- Pasos de instalación:

- [Paso anterior-1] Instalar herramientas de compilación y construcción.
 - gcc, g++, autotools, cmake, scons, ...
- [Paso anterior-2] Instalar dependencias.
 - Bibliotecas externas (.so, .a) y otras herramientas.
- [Paso 1] Descargar el software (formato .tar.gz, .tar.bz2).
 - cd /opt/prebuilds && wget http://www.python.org/ftp/python/2.7.6/Python-2.7.6.tgz
- [Paso 2] Descomprimir.
 - tar -xzf Python-2.7.6.tgz
- [Paso 3] Leer README/INSTALL. Preconfigurar los Makefiles (rutas) y resolver las posibles dependencias (software previo).
 - cd /opt/prebuilds/Python-2.7.6/ && ./configure --prefix=/usr/local/
- [Paso 4] Compilar el paquete e instalarlo en un directorio diferente (/usr/local/).
 - make -j <num cores> && make install

- **DCVS Sistem:**

No todo el software libre se puede obtener mediante ficheros .tar, también se obtienen de plataformas como git o mercurial.

Instalación desde Paquetes:

- Un paquete de software contiene el código fuente o binarios compilados y los script de pre/post instalación.

Ventajas:

- Administración unificada y organizada del software instalado (base de datos).
- Simplifica el proceso de instalación (no se requiere compilador, pre/post instalación, etc.).
 - Es decir, no me tengo que preocupar de la instalación completa. Hay una gestión de la información asociada al software que me instalo que no me hace falta realizar.

- Pueden ser de 2 tipos: binario o paquete de código fuente.
- Se utiliza el comando **dpkg** para el manejo de paquetes en debian (.deb):
 - ➔ **Presenta un gran PROBLEMA:** No gestiona las dependencias automáticamente y es una acción que manualmente es muy tediosa.

Instalación desde el Repositorio:

➔ Para evitar la instalación manual de todas las dependencias de un programa:

- **APT**: Advanced Packaging Tool:
- Automatiza la instalación de dependencias y el mantenimiento de nuevas versiones.
 - o **Apt-get <option>**: Para actualizar la base de datos local para tener la lista de paquetes de instalación actualizados a su última versión, realizamos "apt-get update".
(Esta acción se recomienda siempre realizarse antes del apt-get install.)
 - o **Apt-cache <option>**: Con la opción de "search" puedo encontrar como se llama el paquete que me tengo que instalar de un programa que he editado su código para que me funcione e instale.
- Confundir versiones en la instalación de paquetes puede dar errores.
Siempre tener claro que distribución estoy utilizando: en **/etc/apt/sources.list**
 - o deb -> ya compilado con ejecutable.
 - o deb-src -> código fuente pendiente de instalación.

RESUMEN INSTALACIONES:

1. **Instalación desde Código fuente:**
 - a. Cuatro pasos, requieres compilar y gestionar las dependencias.
 - b. Es una instalación "manual".
2. **Instalación desde Paquetes:**
 - a. Paquete software con los binarios compilados y scripts pre/post instalación, o bien con el código fuente.
 - b. Comando dpkg
 - i. Requiere gestionar las dependencias manualmente, lo que es muy tedioso.
3. **Instalación desde Repositorio:**
 - a. Evita instalación manual de todas las dependencias.
 - b. Comando APT.
 - i. Automatización de todos los pasos de la instalación software.

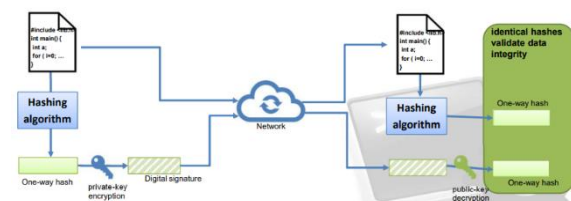
Actualización Debian:

- Ninguna actualización del sistema es infalible:
 - o Discute, prepara y prueba cualquier proceso adecuado de conmutación por error o recuperación.
- Actualiza completamente la distribución actual:
 - o apt-get update; apt-get upgrade; apt-get dist-upgrade
- Busca posibles fuentes de inconsistencias:
 - o Verificación de integridad y consistencia de la base de datos (dpkg -C).
 - o Comprueba qué paquetes se han retenido y no se han actualizado (apt-mark showhold).
- Actualiza el repositorio de paquetes:
 - o Edita /etc/apt/sources.list para incluir los nuevos repositorios de paquetes.
- Actualizar al nuevo sistema:
 - o apt-get update; apt-get upgrade; apt-get dist-upgrade
- Verifica tu nuevo sistema antes de ponerlo en producción.

Seguridad:

La **criptografía asimétrica** consta de varias llaves.

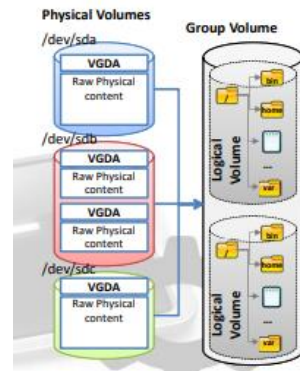
- ➔ Una llave privada y otra pública, la privada encripta y la pública asociada a esa privada es capaz de desencriptar lo que la otra llave encriptó.
 - o La llave pública se distribuye en nuestros equipos pero la privada se queda en el repositorio.
- ➔ De esta manera, si los códigos hash no son iguales significa que han habido modificaciones en el código, y por lo tanto podría haber algún virus o fallo.
- ➔ Por esto se realiza la validación por hashes iguales.



Practica 7: Advanced File Systems (Storage)

Logical Volume Manager (LVM):

- Crea una línea de abstracción sobre el almacenamiento físico, permitiendo la creación de almacenamiento lógico.
 - o Es decir, permite gestionar todo el almacenamiento del que disponemos y distribuirlo y utilizarlo como queramos.
- Jerarquía de LVM
 - o **Volúmenes físicos (PV):**
 - El nivel más bajo de la jerarquía de LVM.
 - Pueden ser discos completos, particiones o dispositivos RAID.
 - Contienen el VGDA (Área de Descriptores del Grupo de Volúmenes) y el contenido físico en bruto.
 - o **Grupos de volúmenes (VG):**
 - Equivalentes a "superdiscos". Se construyen con uno o más PV.
 - Se pueden agregar más PV al GV sin modificar los anteriores.
 - o **Volúmenes lógicos (LV):**
 - Equivalentes a "superparticiones". Los sistemas de archivos se crean en un volumen lógico.



Administración de LVM:

- Comando **pvcreate** (PV): creación de un Volumen Físico.
 - o Sintaxis: pvcreate [partición/disco completo/dispositivo RAID]
- Comando **vgcreate** (VG): creación de un Grupo de Volúmenes a partir de varios PVs.
 - o Sintaxis: vgcreate [nombre-vol] [lista de PVs]
 - o Ejemplo: vgcreate vg01 /dev/sdb /dev/sdc1 (agrupar disco sdb y partición sdc1 en un GV).
- Comando **lvcreate** (LV): creación de un Volumen Lógico.
 - o Sintaxis: lvcreate [GV] -L[tamaño] -n[nombre-lv]
 - o Ejemplo: lvcreate vg01 -L1000M -nvol1 (después de esto, se puede crear el sistema de ficheros con mkfs).
- Nota: Windows no soporta LVM, así que no será posible acceder a ninguna partición LVM desde windows.

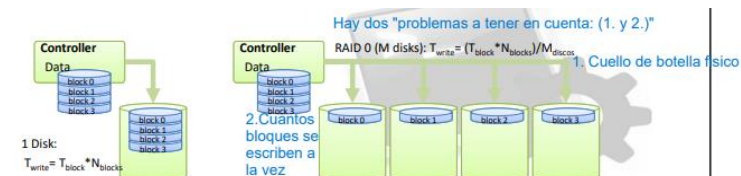
RAID:

- Conseguir mejorar las prestaciones y fiabilidad de los discos haciendoles tolerantes a fallos.
- Transparente al SO si es un raid hardware, el raid software si es visible.
- **RAID 0:**

- o El objetivo es mejorar el rendimiento de lecturas y escrituras.
- o Utiliza el 100% del almacenamiento disponible.

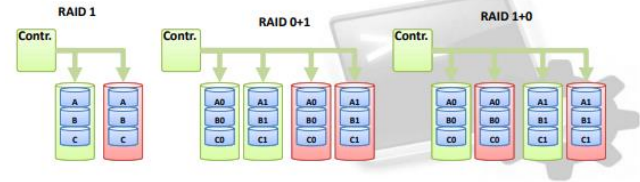
- Reparte los bloques a lo largo de todos los discos disponibles, permitiendo el acceso paralelo a los discos.

- o No tolera fallos.
- o A simple vista, realizando operaciones secuenciales, como transferir un fichero de gran tamaño, los dos discos implementando raid 0 en teoria es el doble de rapido que un solo disco de gran tamaño.
 - Aun asi, en la implementacion práctica puede ocurrir que los tiempos se normalicen y no hayan tantas diferencias.



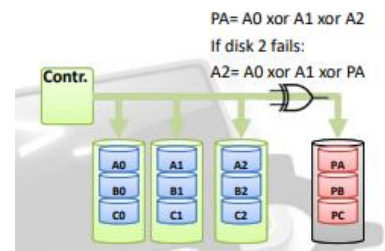
- RAID 1:

- El objetivo es mejorar las lecturas.
 - Tener dos copias completas de la información, para que si quiero leer el bloque A y el B, puedo leer de un disco el A y a la vez de otro disco el B.
- Se aplica una gran redundancia, por cada disco existe otro con el mismo contenido.
 - Por esto mismo las escrituras son más lentas.
 - La capacidad disponible es del 50%.
 - Permite que un disco falle.



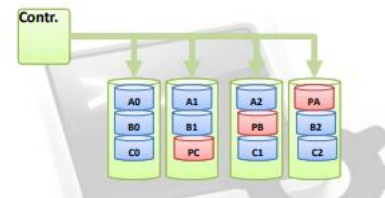
- RAID 4:

- Un disco almacena información de la paridad del resto de discos.
- Mejora el rendimiento de lecturas.
- Las escrituras sufren un cuello de botella, al tener que modificar el disco de paridad.
 - Al necesitar escribir en todos los bloques de paridad, se realiza una escritura secuencial que ralentiza el rendimiento del sistema.
- Tolerar un fallo de disco, si un disco falla, mediante el disco de paridad se podrá recuperar.
 - En el caso de que un disco falle, el rendimiento va a sufrir ya que el coste de recuperación es bastante elevado al necesitar la lectura de muchos bloques mas el de paridad, y todo esto mientras el sistema sigue haciendo un uso de los discos para otras operaciones necesarias.
- Solo se pierde un disco del almacenamiento total.



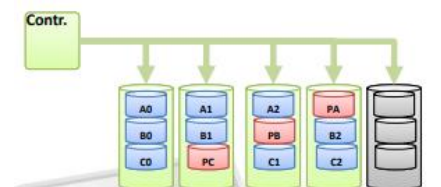
- RAID 5:

- La información de paridad de los discos está repartida entre todos los discos y no en uno solo como ocurre en RAID 4.
 - De est manera ya no ocurre la escritura secuencial que se podría dar en RAID 4 y por lo tanto ya no existe el cuello de botella en las escrituras.
- Tolerar un fallo de disco.
- Solo se pierde un disco del almacenamiento total.



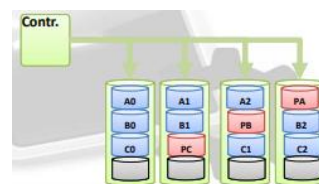
- RAID 5 + Spare Disk:

- Añade un nuevo disco vacío con el fin de que sea utilizado en caso de un fallo de un disco.
 - Automatizan la reconstrucción de los fallos.
 - En caso de fallo, en base a los valores del resto de discos y los bloques de paridad, puedo reconstruir los bloques necesarios para recuperar la información perdida en el fallo en el disco en desuso.



- RAID 5e:

- No añade ningún disco adicional, sino que guarda un bloque libre en cada disco presente en el sistema.
 - De manera que si vuelve a ocurrir un fallo que requiera una escritura de todos los bloques que suponga una escritura secuencial, la evitemos y mejoremos el rendimiento cuando se intente recuperar de un fallo.

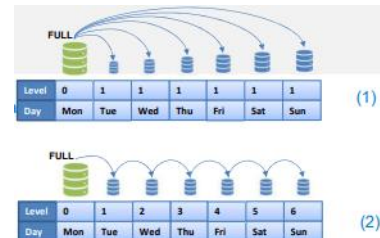


RAID + LVM:

- Con **raid** doy rendimiento y fiabilidad, y con **lvm** rompo las restricciones que tiene raid y podré reparticionar como yo desee los discos.
- Primero se crean las estructuras raid y luego se utiliza LVM.

Backups:

- RAID + journaling no es suficiente para proveer con una disponibilidad del 100%.
 - o Por lo que se realizan backups.
- El backup no incremental(1) guarda info ya asegurada, pero aumenta la eficiencia de recuperacion de datos de backups lejanos a la fecha actual.
- El proceso de recuperacion en el backup incremental(2), restaurar un error que tienes el lunes por ejemplo, siendo viernes, es muy costoso al tener que "iterar sobre los anteriores backups", pero el proceso de guardado es menos costoso cuando se realiza al no tener que guardar tanta cantidad de datos.



Tema 8: The Linux Kernel

El kernel es una parte fundamental del sistema operativo que actúa como intermediario entre el hardware y el software.

- Es responsable de administrar los recursos del sistema:
 - o Memoria
 - o Procesador
 - o Dispositivos de entrada y salida (E/S)
- Proporciona una interfaz para que las aplicaciones puedan acceder a estos recursos de manera segura y eficiente.
- Controla el inicio y apagado del sistema, gestiona la asignación de memoria a los programas en ejecución, administra los procesos y la planificación de la CPU, permite la comunicación entre los diferentes componentes del hardware, gestiona los controladores de dispositivos y proporciona servicios básicos, como la gestión de archivos y el acceso a la red.

Existen diferentes tipos de kernels:

1. Kernel monolítico: En este tipo de kernel, todas las funciones del sistema operativo se ejecutan en un solo espacio de memoria y se comparten entre los diferentes componentes.
2. Kernel microkernel: En este tipo de kernel, solo se implementan las funciones básicas del sistema operativo en el núcleo, como la gestión de memoria y la planificación de la CPU. El resto de los servicios se ejecutan como procesos separados, llamados servidores.

¿Cómo podemos ajustar el kernel?:

- **Estáticamente:**
 - o Código fuente + Compilador + ...
- **Dinámicamente:**
 - o A través de módulos kernel LKM o parámetros de /proc.

Configuración Estática:

- Paso 1: **Obtener el código fuente del kernel:**
- Paso 2: **Configuración:**
 - o Hay que crear un archivo de configuración .config.
 - Existen 2 formas de crear el archivo de configuración:
 1. Scratch: Se realiza manualmente.
 - a. Configuración complicada, ya que se deben responder muchas preguntas y es muy tedioso.
 2. Adapt: Se adapta un fichero de configuración ya existente.
- Paso 3: **Compilación:**
 - o Este paso se divide en:
 1. Compilamos el kernel → make
 2. Compilamos los módulos del kernel → make modules
- Paso 4: **Instalación:**
 - o Este paso se divide en:
 1. Instalamos los módulos del kernel → make modules_install
 - a. Copia la imagen del kernel y la renombra en el directorio deseado.
 2. Crea una nueva entrada en el grub del nuevo kernel → make install

Configuración Dinámica:

- A través del **directorio /proc** y/o los **módulos kernel LKM**.

 - El **directorio /proc** representa el estado del kernel.
 - El directorio **/proc/sys** representa información de los dispositivos, drivers y algunas funciones del kernel.
 - Para configurar los parámetros del kernel en tiempo de ejecución → comando **sysctl**
 - Los cambios realizados que quieran hacerse permanentes, deberán ser representados en el fichero **/etc/sysctl.conf**

 - **Módulos kernel (LKM):**
 - Un módulo es un fragmento del código del kernel que se saca de él.
 - Permite añadir contenido al kernel mientras está en funcionamiento, evitando su recompilación.
 - Si hay problemas en la compilación se sabe o se localiza más fácilmente el fallo al estar trabajando sobre un módulo.
 - Desarrollo y mantenimiento más sencillo al no tener que reiniciar ni reconstruir.
- ➔ Para trabajar con un kernel, normalmente se querrán añadir o modificar drivers, por lo que en vez de realizar los cambios y recompilar el kernel, trabajaremos directamente sobre módulos del kernel para no recompilar el kernel por completo.
- Los módulos se encuentran en el directorio **/lib/modules/moduloEjemplo.ko**
 - Administración LKM:
 - Pueden haber módulos que necesitan de otros módulos para trabajar.
 - Por lo que hay que ser consciente de estas dependencias.
 - Con el comando **ismod** no se gestionan estas dependencias por lo que habría que utilizar otro tipo de comandos.
 - Estas dependencias se declaran en los ficheros generados **.dep**.
 - Para solucionar estas dependencias se utiliza el comando **modprobe** para gestionarlo de manera automática.
 - Carga y descarga de módulos automática:
 - Un LKM puede ser cargado automáticamente cuando el kernel lo necesite mediante el **kernel module loader**.
 - Aun así, el directorio **/etc/modules** mostrará los módulos que deberán ser cargados manualmente en tiempo de ejecución.
 - Instalación de nuevos módulos: Los programadores suelen proveer de módulos con una automatización de instalación.
 - Desde su código fuente.
 - Kernel patch.
 - Script

RESUMEN CONFIGURACION KERNEL:

- El kernel se puede ajustar de 2 maneras:
 1. **Estáticamente** → Creando fichero **.config** → Manualmente(scratch) o adaptando uno existente(Adapt).
 2. **Dinámicamente** → En plena ejecución → A través del fichero **/etc/sysctl.conf** con el directorio **/proc** o mediante módulos kernel LKM.

Sistema de ficheros sysfs:

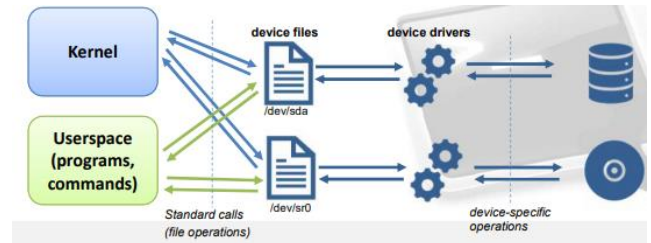
- Una vista similar a un sistema de ficheros de la información y configuración que el kernel proporciona.
- La información se muestra en el directorio **/sys**

Device Driver Module:

- Todo dispositivo tienen asociado al driver un fichero(device file), con el que se gestiona su funcionamiento.
- El objetivo es que tanto el kernel como los dispositivos, utilicen una API similar, una serie de llamadas similares que estarán definidas en ficheros(device files).
- El contenido de estos ficheros lo interpretará el driver.
- Estos ficheros los identificaremos de manera que aparecen estas características que designan que son device files:

```
ls -l /dev/sda  
brw-r----- 1 root root 8, 0 Mar 10 2006 /dev/sda
```

- Estos ficheros de dispositivos se encuentran en el directorio **/dev**
- Estos ficheros se pueden crear de 2 formas:
 - o Manualmente: **comando mknod** (Es insusual)
 - o Automáticamente: mediante el **servicio udev**



Tema 9: Linux Networking

Protocolo "Suite", un conjunto de protocolos diseñados para implementar redes de interconexión.

– Origen: proyecto de investigación del departamento de defensa de EE.UU. (ARPANET). Múltiples componentes, organizados jerárquicamente (pila)

Los datos viajan por la red en forma de paquetes, ráfagas de datos con una longitud máxima impuesta por la capa de enlace.

Cada paquete consta de un encabezado y un payload:

- Encabezado: incluye información de origen-destino y protocolo.
- Payload: la información (Data).

A medida que un paquete viaja por la pila de protocolos TCP/IP, cada protocolo agrega su propio encabezado información

Network interface:

Host / Interfaz:

- Los hosts son computadoras/sistemas individuales.
- Cada host puede tener una o más interfaces de red (NIC) (Cable + wifi)

Cada interfaz representa una conexión a una red diferente (diferente IP).

Linux no realiza gestión de red a través de archivos de dispositivos.

- ethX no tiene un archivo de dispositivo asociado (/dev/ethX no encontrado)
- Las NIC se administran a través de módulos del kernel (controladores)

Network Interface Configuration:

Ifconfig -a ó ip link show Enseñan las diferentes interfaces disponibles

Configuración (Debian): archivo /etc/network/interfaces

- Establece la configuración de las interfaces de red.
- Permite funcionalidades adicionales: rutas, alias, pre/post operaciones,...

– Campos:

- auto <interfaz>: activa la interfaz cuando se inicia el sistema
- iface <interface> <ip_addressing> <method>: configuración de la

interfaz

```
→ vi /etc/network/interfaces
→ allow-hotplug enp0s8
→ iface enp0s8 inet static
→     address      10.0.3.4
→     netmask      255.255.255.0
→     broadcast    10.0.3.255
→     gateway      10.0.3.1
→ ifdown enp0s8
→ ifup enp0s8
```

DHCP (Protocolo de configuración dinámica de host): el servicio DHCP realiza la configuración de red automática para el sistema. Es mas fácil, dinamico y mas seguro. Se requiere un servicio de cliente en cada host

– Cómo especificar que queremos usar DHCP:

- En /etc/network/interfaces:
- man dhclient
- ifconfig eth0 up

La configuración de la interfaz se puede modificar en un sistema "en funcionamiento".

– PASO 1, Quitar la interfaz. (ifdown)

– PASO 2, Modificación. Edite el archivo /etc/network/interfaces o comando ifconfig

– PASO 3, Reiniciar. (ifup)

• Comandos ifup/ifdown (o ip link): encender/apagar una interfaz de red. – Sintaxis: ifdown enp0s3 (apagar tarjeta enp0s3).

• Comando ifconfig (o dirección ip): configuración de parámetros de red. – Sintaxis: ifconfig <interfaz> <dirección> <opciones>

• Ejemplo: ifconfig enp0s3 192.168.1.13 netmask 255.255.255.198 broadcast 192.168.1.191 up

• Ejemplo: dirección IP agregar 192.168.1.13/255.255.255.198 dev enp0s3

• ifconfig -a imprime información sobre las interfaces disponibles.

- ¡¡Precaución!! los cambios realizados con ifconfig no son permanentes (no modifiqué el archivo de interfaces).

Link Layer

El nivel físico en TCP/IP, casi siempre una red ethernet.

– Cada interfaz (NIC) tiene una dirección MAC única.

– Capa a cargo de la conversión de Trama IP <—> Trama Ethernet.

– Protocolo ARP:

• Busca @MAC correspondiente a una @IP en la tabla ARP local (caché de direcciones traducidas)

• Si no está en la tabla, realiza un broadcast e informa al receptor. La tabla ARP se actualiza para futuras conexiones

– Comando arp (ip neighbour): manipulación/visualización de la tabla ARP.

– Configuración/Modificación de @MAC: • # ifconfig eth0 hw ether 00:02:B3:19:C8:21

Network Layer

A través de ARP solo se pueden alcanzar hosts en mi segmento de red. Se deben establecer rutas IP para direcciones externas.

Tablas de rutas: información sobre cómo llegar a los destinos IP

– Destino: identifica la red de destino.

– Gateway: cómo llegar al destino (* significa que el paquete ya está en esa red)

– Genmask: máscara de red (identifica la subred).

– Iface: interfaz de red para llegar a la red de destino.

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.10.0	*	255.255.255.0	U	0	0	0	eth1
127.0.0.0	*	255.0.0.0	U	0	0	0	lo
default	192.168.10.1	0.0.0.0	UG	0	0	0	eth1

Configuración manual de tablas de rutas: Comando route (o ruela ip)

- #route -n : muestra tablas de rutas.
- # route add -net 192.168.1.0 netmask 255.255.255.0 eth0 agrega una ruta para un segmento de red
- # route add default gw 192.168.1.1 eth0 agrega ruta predeterminada a otras subredes

Rutas dinámicas (automáticas)

– La configuración estática de las tablas limita su funcionalidad.

- Válido para redes estables (no muy grandes...)
- Requiere conocimientos sobre topología de red.

– Entornos complejos: Rutas Dinámicas (Daemon “routed” o “gated”. OSFP, RIP, BGP,...). Quizás uno de los aspectos más complejos en cuanto a la administración de la red.

Resolución de nombres: traducción de nombre<->IP, la agenda de la red:

– Opción 1: a través del archivo /etc/hosts: Razonable para redes pequeñas y privadas. No sirve para el resto de casos. Agregar un nuevo host requiere modificar todos los archivos /etc/hosts en la red.

– Opción 2: Servicio de Nombres de Dominio (DNS) Servidor dedicado encargado de realizar la conversión. Cada host debe estar configurado para hacer uso de su servidor de nombres correspondiente. El cliente se configura a través del archivo /etc/resolv.conf

el contenido de /etc/resolv.conf también se puede crear a través de /etc/network/interfaces: (se requiere el paquete resolvconf)

```
iface enp0s3 inet static
    dns-nameservers 1.1.1.1 2.2.2.2
```

```
search localdomain
search atc.unican.es unican.es

nameserver 193.144.193.11
nameserver 193.144.193.22
nameserver 192.168.0.105
```

Checking Network Status

1. Verifique la interfaz: – Eche un vistazo a la definición: /etc/network/interfaces

- Edite el archivo y corrija posibles errores, luego reinicie (ifdown/ifup, ifconfig [dev] down/up)
- Verifique el estado de la interfaz: ifconfig

2. Verifique la capa de red:

- Verifique la tabla de rutas (check del gateway) Edite la tabla con el comando de ruta si es necesario.
- Comprobar la resolución de nombres

- /etc/hosts
- /etc/resolv.conf (el DNS está correctamente definido)

3. Siempre verifique el estado: haga ping a alguna dirección conocida

Del colegón:

Aquí tienes los pasos para realizar la verificación del estado de la red en un sistema Linux:

1. Verificar la interfaz:

- Abre el archivo de configuración de la interfaz de red utilizando un editor de texto. En este caso, se menciona el archivo `/etc/network/interfaces`. Puedes utilizar el comando `sudo nano /etc/network/interfaces` para editarlo.

- Verifica si hay posibles errores en la configuración de la interfaz y corrígelos si es necesario.
- Reinicia la interfaz de red para aplicar los cambios utilizando los comandos ``sudo ifdown [nombre de la interfaz]`` y luego ``sudo ifup [nombre de la interfaz]``.
- Verifica el estado de la interfaz ejecutando el comando ``ifconfig`` o ``ip addr show``.

2. Verificar la capa de red:

- Verifica la tabla de rutas utilizando el comando ``ip route`` o ``route -n``. Comprueba que la configuración de la puerta de enlace (gateway) sea correcta. Si es necesario, puedes editar la tabla de rutas utilizando el comando ``sudo ip route add default via [dirección IP de la puerta de enlace]``.
- Comprueba la resolución de nombres:
 - Verifica el archivo ``/etc/hosts`` para asegurarte de que las entradas de nombres estén configuradas correctamente. Puedes editarlo con el comando ``sudo nano /etc/hosts`` si es necesario.
 - Verifica el archivo ``/etc/resolv.conf`` para asegurarte de que la configuración del servidor DNS sea correcta. Puedes editarlo con el comando ``sudo nano /etc/resolv.conf`` y asegúrate de que la línea "nameserver" tenga la dirección IP correcta del servidor DNS.

3. Siempre verifica el estado:

- Realiza un ping a una dirección conocida para comprobar la conectividad de red. Puedes utilizar el comando ``ping [dirección IP o nombre de dominio]`` para hacerlo. Por ejemplo, puedes hacer ping a la dirección IP de un servidor web conocido o a un nombre de dominio como "google.com".