Tema 4. Sistemas Operativos (Parte 3)

SISTEMAS INFORMÁTICOS. 1º DAW.

Índice:

1. Preámbulo

Antes de instalar un nuevo SO debemos decidir dónde se va a ubicar:

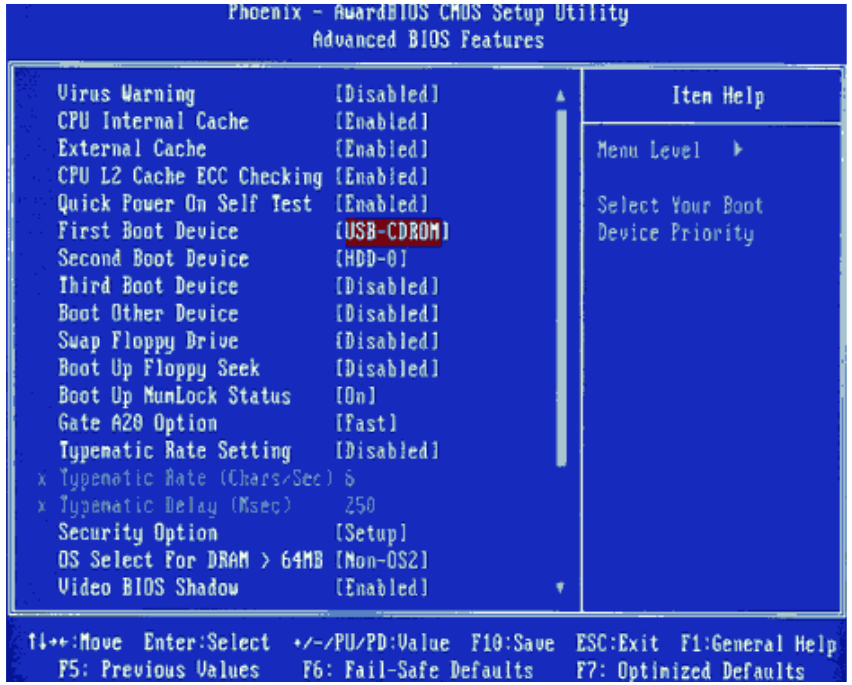
- Se instalará como único SO en el ordenador.
- Convivirá con otro SO, de forma que durante el arranque se pregunte en cuál de ellos se desea iniciar sesión.

Las siguiente decisión es establecer el **número y tamaño de las particiones** a utilizar y el **sistema de archivos** con el que estarán formateadas.

1.1. Secuencia de arranque

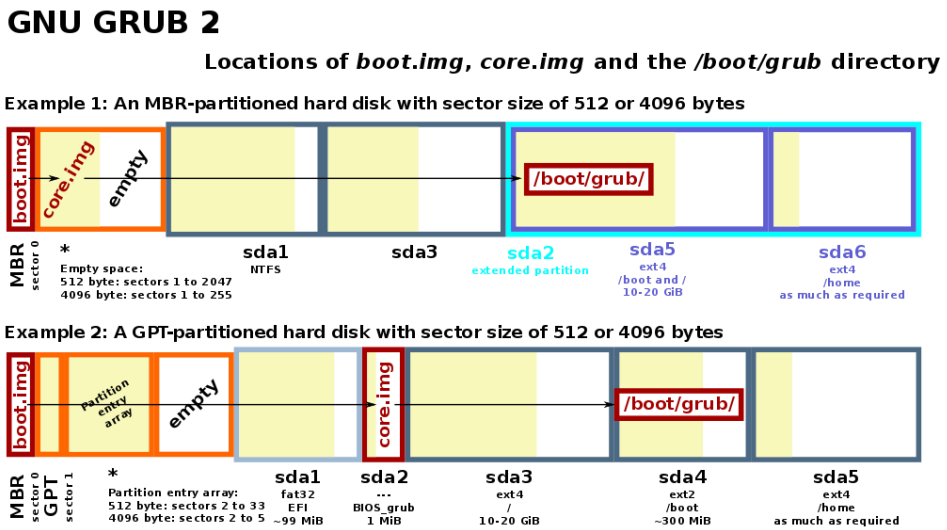
La secuencia de arranque (*boot*) son los pasos que sigue el ordenador desde que se pulsa el botón de encendido hasta que el SO elegido toma el control:

1. La CPU comienza apuntando a una ROM, que contiene un firmware denominado BIOS (antiguo) / UEFI (moderno) y ejecuta una secuencia POST (*Power-On Self- Test*).
2. Al final de la secuencia, el firmware se dirige, en orden, a los dispositivos de arranque configurados (HDD, SSD, Pendrive, CD, red) en busca del Sistema Operativo.
3. Al comienzo del dispositivo de almacenamiento secundario escogido debe existir un sector o bloque de arranque, donde se localiza el Gestor de arranque.



1.1. Secuencia de arranque

4. El Gestor de arranque consulta la tabla de particiones para decidir en qué partición buscará el SO. Una versión sencilla es el “boot loader”, que tiene ya establecida cuál es la partición en la que buscará el SO. Las más avanzadas (ej. *Grub*) ofrecen al usuario elegir entre las particiones que contienen un SO.
5. Se da paso al SO elegido para que complete su propia carga en memoria.



2. Particiones

Las particiones son “bloques” en los que se divide una unidad física de almacenamiento (HDD, memoria Flash, DVD, etc.).

Cada partición es interpretada como un volumen (und. Lógica) diferente y podrá formatearse con un **sistema de archivos** distinto.

En GNU/Linux tenemos/teníamos al menos 2 particiones: *swap* (intercambio) y sistema (/). Suele existir una 3º partición para la información de usuarios (/home). Existe un único árbol de directorios sobre el que se montan las distintas particiones.

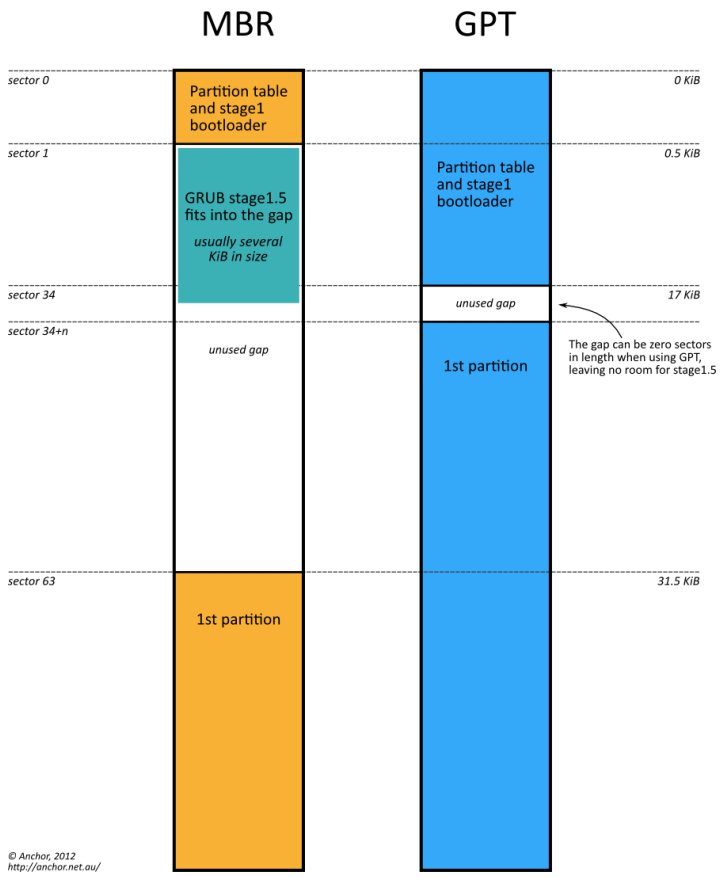
En las versiones más modernas de Windows, existe una partición específica del sistema y al menos una con el sistema y al/los que se le/s asigna/n una letra de unidad (con su propio árbol de directorios).

2.1. Esquemas de particionado

Existen dos tipos de esquemas de particionado:

- MBR (*Master Boot Record*): utiliza un *Registro de Arranque Maestro*. Original de sistemas con BIOS Clásico.

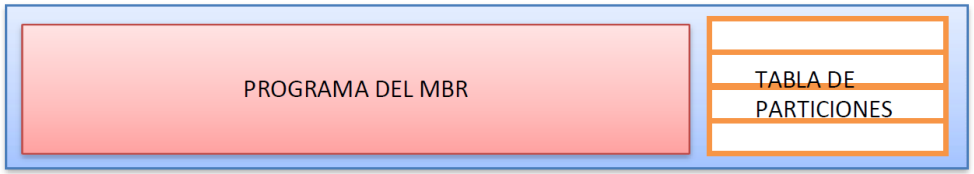
- GPT (*GUID Partition Table*): utiliza una *Tabla de Particiones de Identificador Único Global*. Utilizado en sistemas modernos UEFI, aunque estos también pueden trabajar con MBR en un modo denominado Legacy.



2.1.1. *Master Boot Record*

1º sector de los dispositivos de almacenamiento propios de ordenadores IBM-PC (BIOS). Es un sector de arranque de 512 bytes con la siguiente estructura:

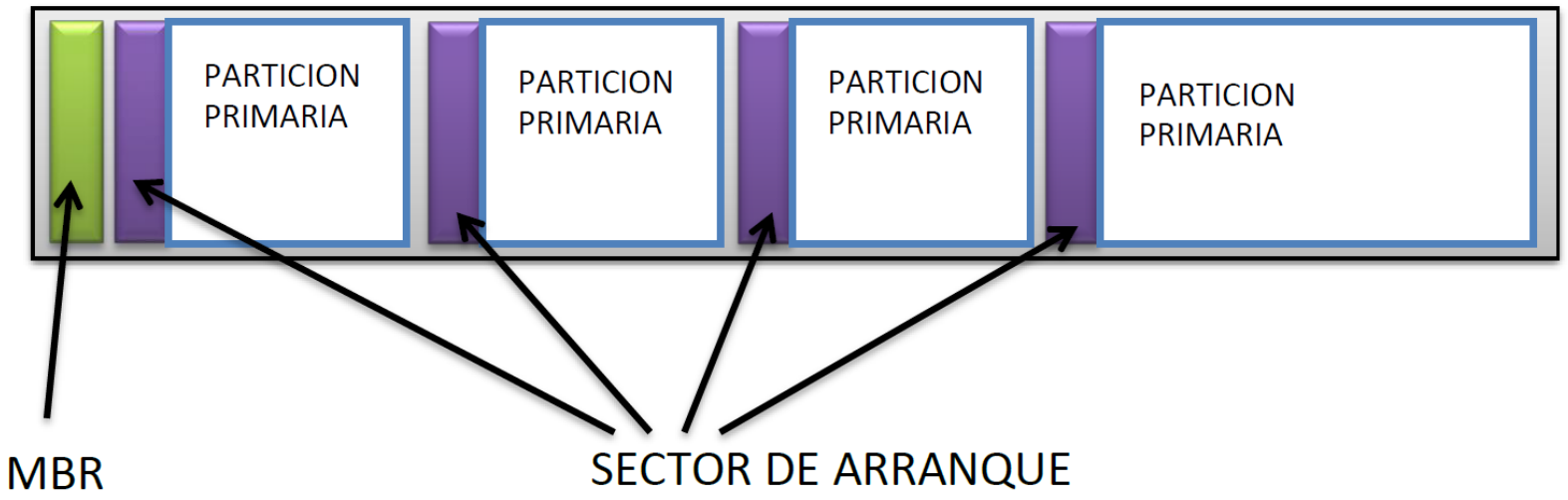
Ubicación en el HDD	Propósito del Código
0-446 bytes	**Código de arranque de MBR**: pequeño programa (binario ejecutable) , señalado por la BIOS, que identifica la partición activa e inicia el proceso de arranque (*boot*), pasándole el testigo al primer sector de la partición activa.
447-510 bytes	**Tabla de particiones**: 4 registros en los que se detalla para cada partición su principio, final, tamaño, nº identificador y un marcador de "activo" (sólo puede haber una partición marcada como activa).
511-512 bytes	**Firma de arranque MBR**, 0xAA55.



1er sector

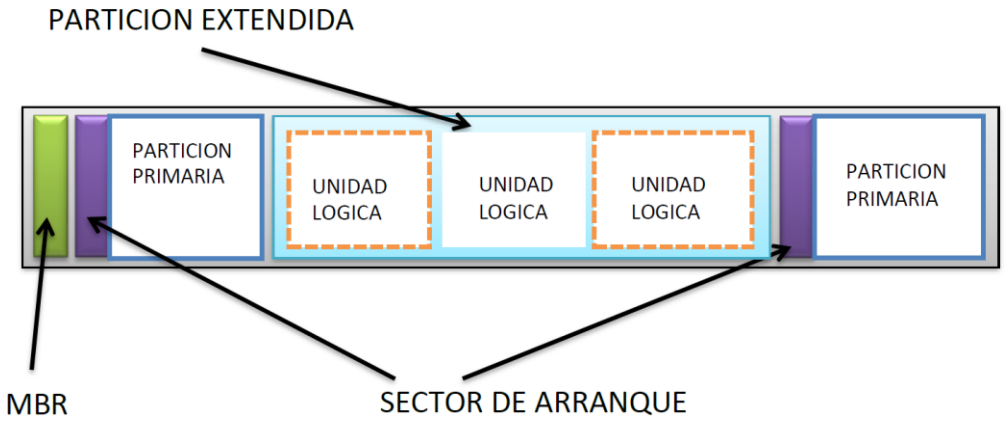
2.1.1.1. Tipos de particiones (con MBR)

- Particiones primaria: en una unidad física puede haber **como máximo 4**. Para que un SO pueda utilizar una unidad de almacenamiento, debe existir en ésta, cómo mínimo, una partición primaria.



2.1.1.1. Tipos de particiones (con MBR)

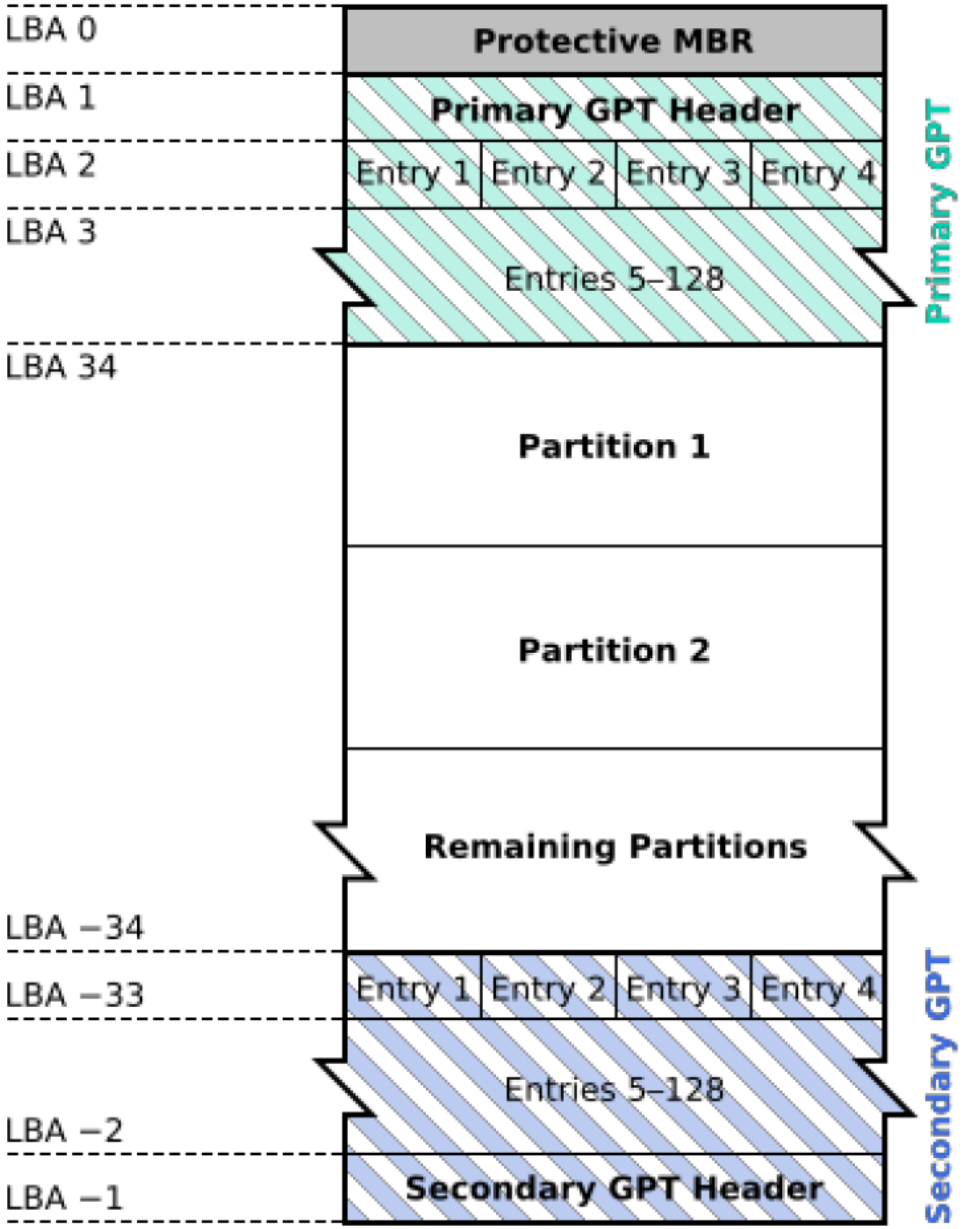
- Partición extendida: en una unidad física sólo puede haber una (o no existir). Ocupa el rol de una partición primaria, por lo que sólo podrá haber otras 3 primarias. Se inventaron para evitar la limitación de nº de particiones primarias.
  - Partición lógica: se ubican siempre dentro de una partición extendida. Pueden definirse hasta un máximo de 23 por partición extendida.
- Espacio no particionado: este espacio no es “utilizable”, el SO “no lo ve”.



2.1.2. *GUID Partition Table*

Apareció para resolver los problemas de MBR (es parte del estándar UEFI).

Ubicación en el disco duro	ropósito
Primer sector lógico del disco (512 bytes)	**Protective MBR**: igual que un MBR normal. Se utiliza por compatibilidad.
Segundo sector lógico del disco (512 bytes)	**Cabecera GPT Principal**: información relacionada con la identificación de disco, ubicación de la tabla de particiones, localización de la 2º cabecera.
16 kiB (por defecto)	**Tabla GPT Principal**: 128 entradas de partición (por defecto, puede ser más), cada una con 128 B. Cada partición tiene un único GUID.
16 kiB (por defecto) antes del último sector lógico del disco	**Tabla GPT Secundaria**: byte por byte idéntica a la tabla Principal. Se utiliza para recuperación en caso de que la tabla principal esté dañada.
Último sector lógico del disco – últimos 512 bytes	**Cabecera GPT Secundaria**: idéntica a la GPT principal. Se utiliza para recuperar información en caso de que la GPT principal esté dañada.


LBA 0
Protective MBR
LBA 1
Primary GPT Header
LBA 2
Entry 1
Entry 2
Entry 3
Entry 4
LBA 3
Entries 5–128
Primary GPT
LBA 34
Partition 1
Partition 2
Remaining Partitions
LBA −34
LBA −33
Entry 1
Entry 2
Entry 3
Entry 4
Entries 5–128
LBA −2
LBA −1
Secondary GPT Header
Secondary GPT

2.1.2.1. Ventajas de GPT

- Proporciona un GUID único para el disco y para cada partición.
- Un nº arbitrario de particiones (dependiendo del espacio asignado por la tabla de particiones). No hay necesidad de particiones extendidas y lógicas.
- Almacena una copia de seguridad del encabezado y de la tabla de particiones al final del disco que ayuda en la recuperación en el caso de que los primeros estén dañados.
- El tamaño máximo del disco manejable es de 2 ZiB.

Ejercicios

14. ¿Cuál es la capacidad máxima de un disco duro con MBR?

15. Cuál es el número máximo de particiones en un disco con MBR?

16. ¿Cuánto ocupa en disco el sistema de particionado GPT?

17. ¿De qué tipo o tipos ha de ser una partición para que aloje un SO en MBR? ¿Y en GPT?

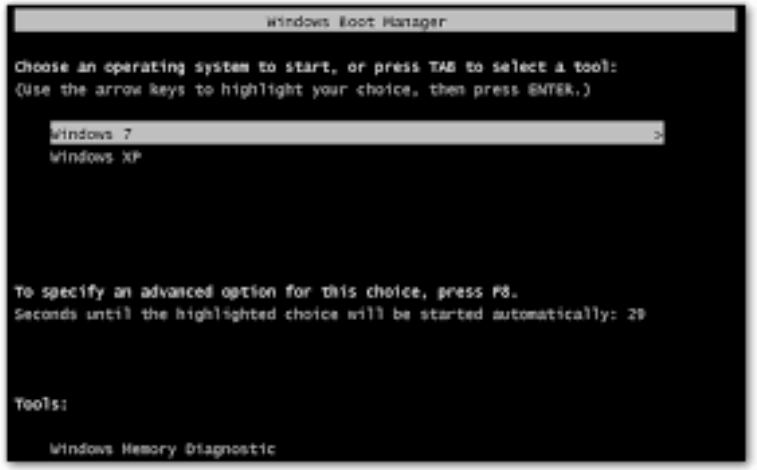
18. ¿Para qué sirve el *Protective MBR* en GPT? (usa tus propias palabras).

3. Gestores de Arranque

Los gestores de arranque son programas que permiten la carga del sistema operativo. Los más simples permiten la carga de un único sistema mientras los más avanzados ofrecen un menú con posibilidad de arrancar distintos sistemas operativos ubicados en distintos discos y particiones.

Entre ellos:

- NTLOADER
- BOOTMANAGER (bootmgr)
- LILO
- GRUB 2



3.1 Windows Boot Manager

1. Se carga y ejecuta el **POST**

2. Se carga el **MBR** del disco duro (si es la opción elegida en la BIOS)

3. Se carga el **sector de arranque** de la partición primaria activa

4. Se carga el programa **bootmgr.**

5. bootmgr ajusta el procesador para trabajar a 32 bits o 64 bits.

6. bootmgr lee la base de datos **BCD** y muestra un menú si es necesario

7. El usuario selecciona un sistema operativo del menú, o se carga por defecto uno de ellos

8. bootmgr carga **winload.exe.**

9. Winload.exe carga **NTOSKRNL.EXE** (Núcleo del sistema operativo o Kernel).

10. NTOSKRNL.EXE lee el **registro** de Windows, y procede a ir cargando el sistema completo.

3.1 Windows Boot Manager

Mientras que el gestor ntldr usaba un fichero de texto denominado boot.ini para configurar sus opciones, bootmgr utiliza una base de datos conocida como Boot Configuration Data (BCD) que no puede ser editada directamente como lo era el boot.ini ya que no es un fichero de texto.

El BCD es una base de datos con datos sobre la configuración del arranque que se suele almacenar en \Boot\Bcd.

Windows dispone de varias herramientas para configurar esta base de datos BCD.

- El comando **bcedit.exe**,
- Utilidades graficas de terceras partes tales como EasyBCD.

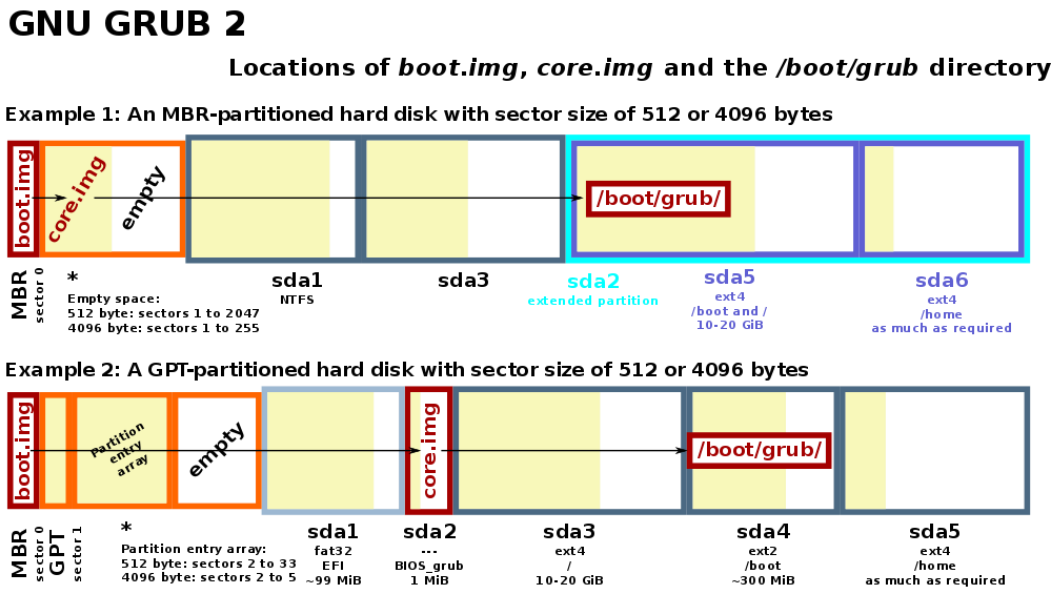
3.1 Windows Boot Manager

Características Importantes de Windows 8 y 10 (UEFI)

- Secure Boot. (Se comprueba la firma digital del sector de arranque).
- Trusted Boot. (Se comprueba la firma digital del kernel de Linux).
- Fast Startup. (El núcleo se hiberna y se carga en cada ejecución).

3.2 GRUB 2

El GRand Unified Bootloader (GRUB) es un gestor de arranque múltiple que se usa comúnmente para iniciar dos o más sistemas operativos instalados en un mismo ordenador. Otros gestores de arranque usados anteriormente en Linux son el syslinux y el lilo.



4. Sistemas de archivos

El sistema de archivos (FS) es el componente del SO que se encarga de organizar el modo en que se guardan los datos dentro de un partición. Conceptos:

- Archivo/fichero: serie de bytes almacenados en un dispositivo de almacenamiento que, en conjunto, forman una unidad lógica. Suele estar identificado por:
  - Nombre: identifica el contenido del archivo.
  - Extensión: identifica el tipo de archivo (documento, imagen, etc.).

¡No pueden existir 2 archivos que tengan el mismo nombre y extensión en la misma ubicación!

- Carpeta/directorio: modo de agrupar archivos para facilitar su organización. Tienen un nombre que las identifica. Para un FS, una carpeta es un archivo que contiene información sobre el modo en que se organizan otros archivos.

¡No pueden existir dos carpetas con el mismo nombre en la misma ubicación!

4. Sistemas de archivos (II)

El FS se encargará de aspectos como…

- Organizar de forma lógica los bloques/clúster del dispositivo para construir archivos y directorios.
- Asignar espacio de almacenamiento a los archivos y mantener el control sobre los bloques que pertenecen a cada archivo.
- Ofrecer los mecanismos que permitan crear nuevos archivos, cambiarles el nombre y/o modificar su contenido o eliminarlos.
- Mantener la estructura jerárquica del sistema de directorios.
- Controlar el acceso seguro a los archivos. Es decir, que sólo puedan acceder a los datos los usuarios autorizados.
- Controlar que bloques permanecen disponibles para ser ocupados en cualquier momento.

4.1. Tipos de sistemas de archivos

Es frecuente que cada familia de Sistemas Operativos tenga sus propios sistemas de ficheros (*FileSystem*).

Existe un gran nº de sistemas de archivos distintos, y vamos a ir desgranándolos según las siguientes familias:

- Familia Microsoft.
- Familia GNU/Linux.
- Familia Apple.
- Otros sistemas de archivos.

4.1.1. Familia Microsoft (I)

- FAT (*File Allocation Table*): utilizado en los primeros SSOO de Microsoft, sin características de seguridad más allá de los atributos de archivo, y éstos se fragmentaban con relativa seguridad. Sus diferentes variantes, en orden cronológico son:
  - FAT12: implementado para el QDOS (1980). Capaz de direccionar un máximo de 32MiB y usaba nombres de archivo con un máximo de 8 caracteres y 3 de extensión. Para MSDOS se utilizó la versión FAT16 (hasta 90 MiB).
  - VFAT: implementado para Windows 3.11,. Usaba el modo protegido de 32 b. del procesador y accedía directamente al HW y la caché (+ velocidad). Una mejora para Windows95 permitía hasta 255 car. de nombre.
  - FAT32: apareció con Windows95 ORS2. Podía direccionar casi 8 TiB en teoría, pero fue limitado primero a 128 GiB y, con XP, a 32 GiB. El tamaño máximo de archivo es de 4 GiB.

4.1.1. Familia Microsoft (II)

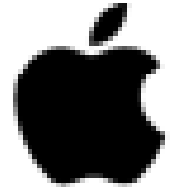
- NTFS (*New Technology File System*):
  - Comparado con FAT, ofrece un mayor rendimiento, seguridad y fiabilidad: puede recuperarse de errores de disco, aplica técnicas de *journaling*, y más.
  - Tiene el inconveniente de que sus estructuras ocupan demasiado para discos pequeños.
  - Apareció con las versiones profesionales de los SSOO Microsoft (Windows NT), aunque se incorporó a los SSOO de escritorio desde Windows XP.
  - Basado en el HPFS, sistema de archivos desarrollado por IBM y Microsoft para OS/2.
  - Permite direccionar volúmenes de hasta 16 TiB (aunque puede llegar a los 16 EiB).

4.1.2. Familia GNU/Linux

Esta familia soporta gran variedad de FS, siendo los más comunes:

- ext2 (*2º Extended filesystem*): introducido en **1993** como evolución de un FS anterior. Admitía hasta 16 TiB, con archivos de hasta 2 TiB y nombres de 256 caracteres.
- XFS: apareció en **1994** sobre IRIX, el UNIX de la empresa Silicon Graphics Inc, pero se liberó en el año 2000. Se añadió al núcleo de GNU/Linux el año 2004. El 1º en ofrecer "journaling". Admite dispositivos de hasta 16 EiB, con archivos que puede llegar a 8 EiB.
- ReiserFS: apareció a principios de **2001** con la versión 2.4.1. del núcleo de GNU/LInux. Soportaba "journaling" y limitaciones de 16 TiB dispositivos, 8 TiB archivos, y 256 c. nombres.
- ext3 (3º Extended filesystem): apareció en **2001** para el núcleo GNU/Linux. Mantenía compatibilidad con ext2, añadía soporte para "journaling" y se apoyaba en un árbol binario balanceado (+ rendimiento). Admitía dispositivos de hasta 32 TiB, y archivos de hasta 2 TiB.
- ext4: se publicó en **2006** con el núcleo GNU/Linux y añade mejoras de velocidad y uso de la CPU. Además, admite dispositivos de hasta 1 EiB, con archivos que pueden llegar a 16 TiB.

4.1.3. Familia Apple



- MFS (*Macintosh File System*): sistema incluido en el Macintosh 128K (**1984**). Soportaba nombres con 255 c. (localizaba los primeros 63). Admitía un tamaño máximo de volumen de 256 MiB, pudiendo tener archivos del mismo tamaño.
- HFS (*Hierarchical File System)*: se desarrolló para el sistema Mac OS y podía usarse en discos duros, disquetes y CD-ROMs. Su principal objetivo era aumentar el rendimiento en dispositivos de mayor capacidad (hasta 2 TiB, y archivos de hasta 2 GiB). Mantiene la restricción de localizar archivos sólo por los primeros 63 caracteres. Reemplazó a MFS en septiembre de **1985**.
- HFS+ (o Mac OS extendido): introducido en 1998. Utiliza técnicas de "journaling", emplea Unicode para los nombres de archivo y directorio, y es capaz de direccionar 8 EiB (y admite archivos del mismo tamaño).

4.1.4. Otros sistemas de archivos

Existen algunos sistemas de archivos que tienen un propósito específico, y que no pueden ser catalogados ni como sistemas de archivos de disco, ni como sistemas archivos de red. Entre ellos, podemos destacar los siguientes:

- "swap": contiene un espacio de almacenamiento en disco destinado a apoyar el funcionamiento de la memoria virtual.
- "cdfs": FS virtual GNU/Linux, permite acceder individualmente a las pistas de audio en un CD.
- "udev" y "devfs": sistemas de archivos virtuales que utiliza GNU/Linux para manejar los archivos almacenados en dispositivos vinculados al directorio "/dev".
- "ftpfs": es el sistema de archivos que ofrece acceso a datos a través de FTP (protocolo).
- "archfs": FS creado en espacio de usuario (FUSE), permite navegar por los repositorios rdiff-backup.
- "nntpfs": FS creado en espacio de usuario (FUSE), permite acceder a noticias en Internet (protocolo NNTP).

4.1.5. Transacciones. Sistemas transaccionales.

Transacción: “acuerdo entre dos partes en la que se produce un intercambio”.

Suele estar formada por varias acciones individuales, que será necesario completar en su cjto. para que la transacción pueda darse por terminada.

Ejemplo de transacción: “comprar pan”…

1. Consultar el precio.
2. Comprobar si llevamos dinero suficiente.
3. Pedir la pieza de pan al vendedor.
4. Recibir la pieza de pan.
5. Entregar el dinero correspondiente.

Si se interrumpe la transacción, debe anularse en su totalidad.

4.1.5. Transacciones. Sistemas transaccionales (II).

El *journaling*, o registro diario, es el modo de implementar transacciones en un FS.

Se basa en crear un registro (*journal*) donde se anotan los datos necesarios para rehacer las modificaciones que se hayan hecho durante la transacción por si fallara.

Así se evita la corrupción del FS antes cortes eléctricos o fallos inesperados. Tras un fallo, el sistema sólo tendrá que deshacer las transacciones inacabadas para volver a disponer de un sistema de archivos íntegro.

4.1.5.1. Sistemas de archivos más habituales

Nombre	Sistema Operativo	Otros sistemas operativos compatibles
FAT12, FAT16, FAT32	Toda la familia de SSOO de Windows (FAT32 desde DOS 7.1).	Linux y Mac OS.
ExtFAT (versión FAT para memorias USB)	Toda la familia de SSOO de Windows (a partir de XP SP2).	Mac OS X a partir de la versión 10.6.5.
NTFS	Toda la familia de SSOO de Microsoft a partir del NT (server) y XP (desktop)	Linux con núcleo a partir de las versión 2.2 y MacOS X pero sólo en modo lectura.
HFS	Mac OS (con soporte de sólo lectura a partir de la versión OS X 10.6).	Linux.
HFS +	Mac OS desde la versión 8.1.	-
Ext3	Linux.	Mac OS X.
Ext4	Linux con núcleo a partir de 2.6.28.	Mac OS X (extensión ExtFS).
ReiserFS	Linux con un núcleo parcheado	-

4.2. Atributos y permisos

El SO debe tener la capacidad de controlar qué usuario puede acceder a cada uno de sus recursos (directorios, impresoras, conexiones de red, etc.).

Para lograrlo, cada uno de estos recursos suele tener asociada una Lista de Control de Acceso o *ACL*, en la que se relacionan los diferentes usuarios que pueden acceder y bajo qué condiciones (lectura, escritura, ejecución, …).

Un determinado archivo puede tener asociados diferentes atributos, que informan sobre ciertas características del archivo o del modo en el que el SO debe tratarlo. Así, un archivo puede tener atributos como: *directorio, oculto, de sistema, cifrado, etc.*

4.3. Nombres y rutas

A día de hoy, prácticamente todos los FFSS organizan los archivos de forma jerárquica, permitiendo la que creación de un árbol de directorios que facilitan la organización y clasificación de su contenido.

Es frecuente que la parte final del nombre termine en un punto, seguido de 3 o 4 caracteres que indican el tipo de contenido del archivo. De esta forma, el usuario sabe la clase de información que puede encontrar en su interior. Este grupo de caracteres recibe el nombre de *extensión*.

Algunos FFSS (FAT, NTFS) utilizan las extensiones para establecer el tipo de programa que puede interpretar la información que contiene. En otros como GNU/Linux, se utilizan los *metadatos* de los archivos para identificar el tipo de información que almacenan.

4.3.1. Extensiones frecuentes

Extensiones	Contenido
jpg/jpeg, png, gif, bmp	Archivos de imagen de mapa de bits
ai, svg	Archivos de imágenes vectoriales
zip, rar, cab, 7z, tar	Archivos comprimidos
mp3, wav, wma, aac, ac3, flac	Archivos de audio
avi, mov, mpg, mpeg, divx, wmv, rpm	Archivos de vídeo
vdi, vhd, vmdk	Discos virtuales
htm, html, xml	Lenguaje de marcas
doc, odt, ppt, odp, xls, ods	Archivos de paquetes ofimáticos
Exe, msc, com, dll, sys, tmp	Archivos propios el sistema operativo

4.3.2 Ubicaciones/rutas

Para expresar la ubicación de un archivo o directorio se utiliza su ruta (*path*).

Consiste en la lista jerárquica de directorios que representa el "camino" que debemos seguir para llegar hasta el archivo o directorio, siendo el propio archivo o directorio el último elemento referenciado.

Al escribir la ruta, debemos utilizar un carácter que separe cada elemento del siguiente. Este carácter varía según el SO:

- En los SSOO de Microsoft es la barra invertida (\). Ej. *C:\Windows\archivo1.txt*
- En los SSOO de la familia UNIX es la barra inclinada (/). Ej. */usr/archivo2.txt*

Otros caracteres especiales que podemos utilizar en una ruta son los siguientes:

- Un punto (.): hace referencia al directorio en el que nos encontramos (directorio actual).
- Dos puntos (..): hace referencia al directorio al que pertenece el actual (directorio padre).
- Una virgulilla (~): en sistemas de la familia UNIX hace referencia al directorio personal del usuario.

4.3.2.1. Rutas absolutas y relativas

Existen dos modos de escribir rutas:

- De forma **absoluta**: escribir la ruta partiendo del directorio raíz…
  - En sistema UNIX comienza en “root”, “/”. Ej. /home/Alicia/Documentos/informe.odt
  - En sistemas Microsoft comienza por la letra de la unidad. Ej. c:\Usuarios\Alicia\Documentos\informe.odt
- De forma **relativa**: escribir la ruta partiendo del directorio actual…
  - Ej. en sistemas UNIX: ../../Jacinto/Documentos/memoria.odt
  - Ej. en sistemas Microsoft: ..\..\Jacinto\Documentos\memoria2.odt

Las rutas que identifica recursos compartidos en una red suelen hacer uso de la nomenclatura UNC (*Universal Naming Convention*) y tienen la siguiente sintaxis:

*Servidor**RecursoCompartido**ruta**archivo*

4.3.2.2. Caracteres comodín

Un carácter comodín es aquel que tiene la capacidad de representar a cualquier otro carácter o conjunto de caracteres. Suele utilizarse para sustituir parte del nombre de un archivo o de alguno de los directorios que forma su ruta.

Los dos caracteres comodín más usados son:

- El asterisco (*): representa cualquier combinación de caracteres, incluso ninguno. Ejemplo: supongamos que tengo en un directorio los ficheros *mi_imagen.png, imagen.png, imagen1.png, imagen21.png* …puedo hacer referencia a todos ellos escribiendo **imagen*.png*
- El signo de interrogación (?): representa un único carácter. Así, en el ejemplo anterior puede hacer referencia sólo al último archivo escribiendo *imagen??.png*

Así, podemos referenciar un archivo del que no recordamos su nombre completo o referenciar un grupo de archivos que tengan en común parte del nombre.

Ejercicios

19. Tenemos un fichero que se encuentra almacenado en el directorio: D:\Trabajo\2018-2019, y queremos copiarlo a C:\Users\Trabajador\Desktop.

a) ¿Cuál sería la orden en MSDOS usando direccionamiento absoluto?

b) ¿Y si suponemos que estamos en el directorio D:\Trabajo y queremos mover el fichero a D:\Trabajo\2017-2018? (usa direccionamiento relativo).

20. Crea un patrón de nombre de ficheros usando caracteres comodín para el conjunto (*caps sensitive*): gomezR, GomezS, SergioGomez, ricardogomez, gomezJacinto. ¿Y si sólo quisiéramos con “gomezR” y “GomezS”?