

**CONTENIDO**

***PROCESO DE ARRANQUE DE UN SISTEMA OPERATIVO..... 2***

***ARRANQUE INICIAL. POST..... 2***

***ELECCION Y ARRANQUE DEL SISTEMA OPERATIVO..... 5***

***ORGANIZACIÓN LOGICA DEL DISCO DURO. .... 8***

***GPT..... 16***

    Problemas con el MBR. .... 18

    Ventajas de GPT. .... 18

***arranque de windows xp/2000/2003 ..... 19***

***arranque de windows 7/vista/2008/8/10 ..... 20***

***Arranque de Windows 8/10 ..... 21***

    Secure Boot..... 21

    Trusted Boot..... 21

    Fast Startup ..... 21

***Arranque de linux. grub. .... 23***

***Recuperación de errores en el arranque. .... 25***

    Windows XP..... 27

    Windows 7..... 27

    Windows 8/10/2016..... 28

    Linux..... 28

    Problemas de arranque con UEFI y BIOS. .... 28

***Modos de arranque a prueba de fallos. .... 30***

***EVOLUCIÓN HISTORIA DE LOS SISTEMAS OPERATIVOS..... 34***

## PROCESO DE ARRANQUE DE UN SISTEMA OPERATIVO.

Ya hemos visto anteriormente que el hardware, por si solo es totalmente incapaz de realizar ninguna acción, necesita un software que le indique que tiene que hacer. Cuando encendemos un sistema informático, estamos poniendo en marcha hardware, por lo que se necesitan medios especiales para hacer que se cargue un primer software.

### ARRANQUE INICIAL. POST.

En los ordenadores compatibles actuales el proceso de carga de un sistema operativo cualquiera se compone de una serie de pasos que se inician cuando se enciende o reinicia el ordenador. El proceso comienza siempre en el BIOS, y salvando algunas pequeñas variaciones que puede haber en función de cada fabricante de hardware y del propio BIOS, el desarrollo paso a paso de esta secuencia es el siguiente:

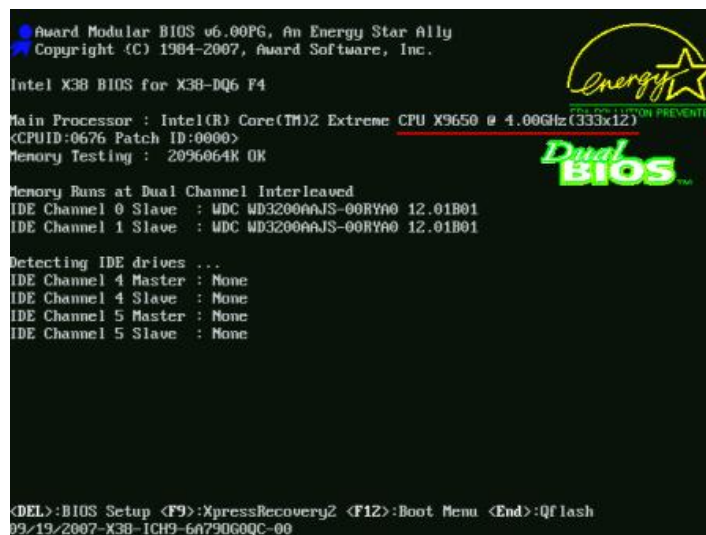


1. Cuando se da tensión a la fuente de alimentación y una vez que la alimentación se estabiliza, genera una señal denominada "**Power Good**" en uno de los cables que va de la fuente de alimentación a la placa base; esta señal es recibida en el juego de chips instalado en la referida placa, y a su vez generan una señal de reinicio (**reset**) al **procesador**. La finalidad de este proceso es evitar que el procesador arranque prematuramente, cuando las tensiones de alimentación no son todavía correctas, lo que podría producir daños en el hardware. Es el mismo sistema que se utiliza para un reinicio en caliente cuando pulsa en el botón marcado "Reset".
2. El procesador arranca cuando se retira la señal de reset. En este momento no existe en su memoria ninguna instrucción o dato, por lo que no puede hacer absolutamente nada. Para salvar este obstáculo, los fabricantes incluyen en la circuitería (hardware) de la placa base un mecanismo especial. El sistema se dirige a una dirección fija de memoria (la FFFF0h en concreto). Esta dirección, situada muy cerca del final de la memoria del sistema en los primeros ordenadores compatibles, es el punto de inicio de la **BIOS**. En realidad este punto de inicio contiene una instrucción de salto (jump) que indica al procesador donde tiene que dirigirse para encontrar el punto donde comienza realmente el programa de carga (**BOOTSTRAP**) de la BIOS. Este programa contenido en esa dirección se lleva a la **CPU** y se ejecuta.
3. La primera parte de este programa de la BIOS inicia un proceso de comprobación del hardware denominado **POST** (Power On Self Test), en caso de existir errores graves, el programa se detiene emitiendo una serie de pitidos (<http://www.bioscentral.com/>) que indican el tipo de error encontrado.



El orden de las comprobaciones del POST depende del fabricante, pero generalmente la secuencia de comprobaciones se resume como sigue:

- a. Comprobación del procesador
- b. Varias comprobaciones sobre la memoria RAM
- c. Inicializar los dispositivos de video y teclado. La comprobación del dispositivo de video incluye cargar y ejecuta la parte de BIOS incluida en el adaptador de video. La mayoría de las adaptadoras modernas muestran en pantalla información sobre sí mismas; es por esta razón por la que, a veces, lo primero que se ve en pantalla es información sobre la propia controladora de video antes que ningún mensaje de la BIOS del sistema.
- d. Determinar el tamaño de la RAM completa y comprobar su funcionamiento (el recuento que se ve en pantalla). Si llegado a este punto existiera algún error en la memoria se mostraría un mensaje de error (el dispositivo de video ya está operativo, así que no hace falta emitir pitidos).



```
Award Modular BIOS v6.00PG, An Energy Star Ally
Copyright (C) 1984-2007, Award Software, Inc.

Intel X38 BIOS for X38-DQ6 P4

Main Processor : Intel(R) Core(TM)2 Extreme CPU X9650 @ 4.00GHz(333x12)
(CPUID:0676 Patch ID:0000)
Memory Testing : 2096064K OK

Memory Runs at Dual Channel Interleaved
IDE Channel 0 Slave : WDC WD3200AAJS-00RYA0 12.01B01
IDE Channel 1 Slave : WDC WD3200AAJS-00RYA0 12.01B01

Detecting IDE drives ...
IDE Channel 4 Master : None
IDE Channel 4 Slave : None
IDE Channel 5 Master : None
IDE Channel 5 Slave : None

<DEL>:BIOS Setup <F9>:XpressRecovery2 <F12>:Boot Menu <End>:QFlash
09/19/2007-X38-ICH9-6A79060C-00
```

- e. Inicializar los puertos: COM (comunicaciones serie), LPT (comunicaciones paralelo), USB, S-ATA, SCSI, etc.
- f. Inicializar, en su caso, el sistema de disquete.
- g. Inicializar el sistema IDE, S-ATA o SCSI. (Discos duros, CDROMS, etc.).

4. A continuación, el BIOS recorre la memoria en busca de la posible existencia de otros programas en ROM para ver si alguno tiene BIOS, lo que ocurre, por ejemplo, con los controladores de disco duro IDE/ATA, cuyos BIOS se encuentran en la dirección C8000h; otros elementos que suelen contar con sus propios BIOS son las tarjetas de red y las controladoras SCSI. Estos módulos son cargados y ejecutados.
5. A continuación, el BIOS muestra su pantalla principal (generalmente con los créditos del fabricante número de versión y fecha). Como hemos visto, el BIOS realiza una especie de inventario del sistema y algunas pruebas para verificar que su funcionamiento es correcto, y en esta pantalla muestra un resumen de los mismos.

```

Phoenix Technologies, LTD
System Configurations
CPU Type      : AMD Athlon(tm) XP   Base Memory   : 640K
CPU ID       : 0681                Extended Memory : 1047552K
CPU Clock    : 2000MHz           L1 Cache Size : 128K
                                           L2 Cache Size : 256K

Diskette Drive A : 1.44M, 3.5 in.   Display Type  : EGA/UGA
Pri. Master Disk : LBA,ATA 100,40022MB Serial Port(s) : 3F8 2F8
Pri. Slave Disk  : LBA,ATA 100,40062MB Parallel Port(s) : 378
Pri. Master Disk : DUD,ATA 33      DDR DIMM at Rows : 2 3 4 5
Sec. Slave Disk  : CHS,PIO 4, 512MB

PCI device listing ...
Bus No. Device No. Func No. Vendor/Device Class Device Class IRQ
-----
0 2 0 10DE 0067 0C03 USB 1.0/1.1 OHCI Controller 10
0 2 1 10DE 0067 0C03 USB 1.0/1.1 OHCI Controller 11
0 2 2 10DE 0068 0C03 USB 2.0 EHCI Controller 5
0 9 0 10DE 0065 0101 IDE Controller 14
0 13 0 10DE 006E 0C00 Serial Bus Controller 10
1 8 0 1106 3043 0200 Network Controller 11
1 9 0 1102 0002 0401 Multimedia Device 11
    
```

En los PC originales la configuración del hardware disponible se efectuaba mediante interruptores ("Jumpers") situados en la placa-base. Hoy en día se utiliza el estándar PnP (Plug and Play), que es capaz por sí misma de detectar y configurar los dispositivos conectados, asignándoles los recursos necesarios y mostrando un mensaje en pantalla por cada uno instalado.

La última instrucción del programa POST se encarga de buscar otro programa que pueda ser cargado en el procesador del PC para que se encargue de seguir arrancando el sistema informático, normalmente cargando ya un sistema operativo.

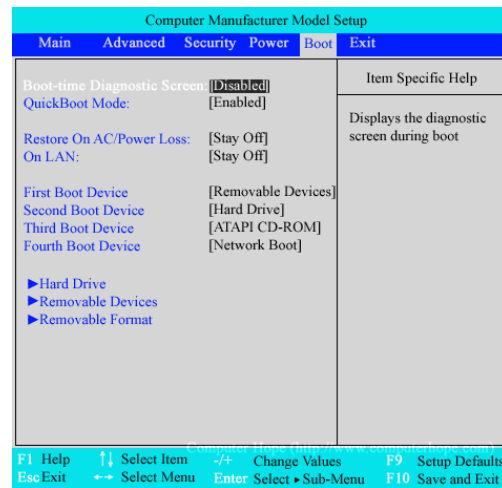
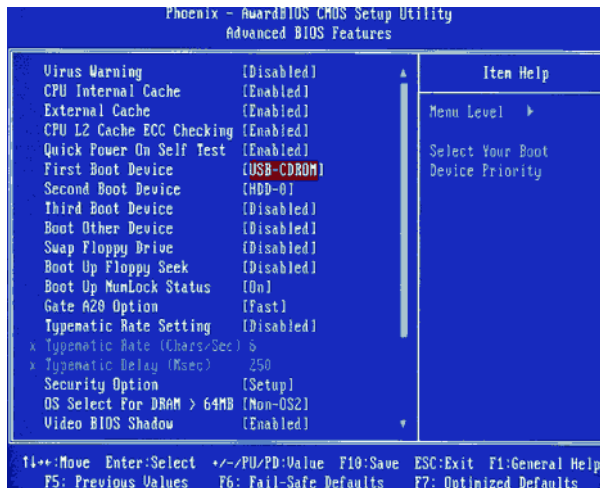
¿Pero dónde buscará el POST el programa a cargar? Y en caso de que existan varios sistemas operativos en varios soportes, ¿cuál de ellos será el elegido?

ELECCION Y ARRANQUE DEL SISTEMA OPERATIVO

En este punto en el que estamos el programa que está en la CPU es el POST, y ya ha concluido todo su trabajo. Pero si dicho programa simplemente liberará la CPU, el equipo se quedaría colgado ya que ningún otro software entraría en el microprocesador. Por ello, la última misión del POST es buscar otro programa, y cargarlo en la CPU antes de liberarla.

En un sistema informático actual podemos tener múltiples discos duros, cada uno de ellos con varias particiones donde pueden estar almacenados varios sistemas operativos, podemos tener un CD en la unidad lectora que también cuente con su propio sistema operativo, podemos tener un disquete de inicio en la disquetera, podemos tener un pequeño sistema operativo en un dispositivo USB, podemos tener un disco duro externo conectado por FireWire; etc. ¿Cómo puede saber el POST a cuál de todos estos programas cederle el control?

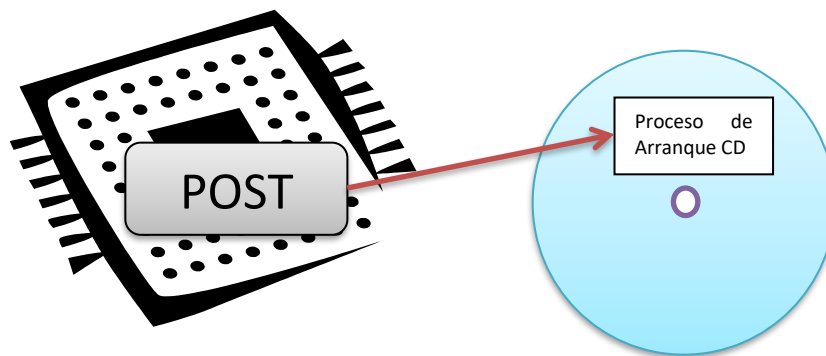
De momento, en la BIOS de casi todos los equipos modernos es posible encontrar unas opciones que indican cual es el soporte de información desde el cual se va a arrancar el sistema (Boot).



Normalmente estas opciones se encuentran en la segunda opción que aparece en el menú de la BIOS (opciones avanzadas de la BIOS o Advanced BIOS Features).

En alguna opción de este menú, normalmente se nos permite indicar varios dispositivos ordenados que utilizaremos para el arranque. Una opción que se puede dejar por defecto, es indicar que se arranque desde el Floppy si existe, luego desde el CD, y por fin del HDD, para que nos permita arrancar el sistema desde disquete, si no existe desde CD, y si tampoco hay ningún CD de arranque, desde el disco duro. En las BIOS más modernas, veremos que también podemos indicarle que arranque desde un puerto USB, desde un puerto SATA, etc.

Si el sistema operativo se ejecuta desde disquete o CD, no hay demasiados problemas, dado que en un disquete o en un CD solo puede haber un único proceso de arranque para un único sistema operativo.

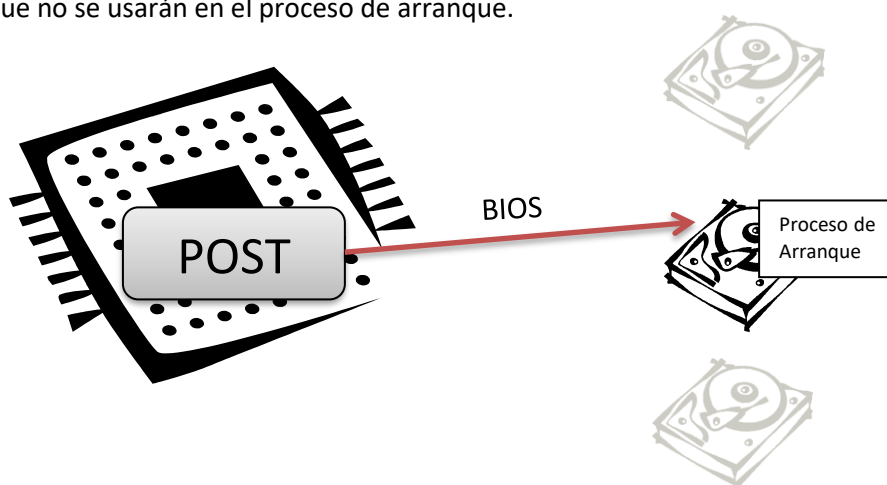


Sin embargo, es posible que en disco duro tengamos varios sistemas operativos para arrancar en nuestra maquina en varias particiones. Además, podemos tener varios discos duros en nuestro sistema, y en cada disco podemos tener varios sistemas operativos instalados.

Desde la BIOS vemos cómo podemos indicar de qué dispositivo queremos arrancar. Podemos indicar normalmente si queremos arrancar desde el disco duro, desde el CD, USB, etc.

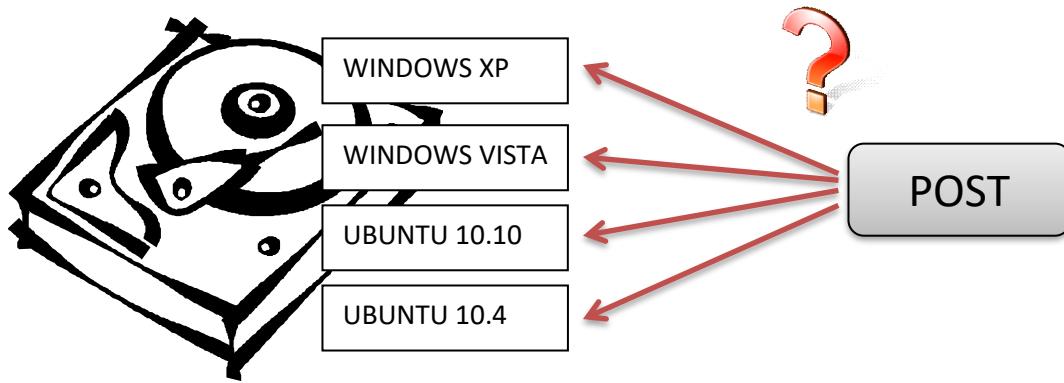
Hay BIOS desde donde se puede indicar incluso desde cuál de los discos duros queremos arrancar (HDD-0, HDD-1, etc.) Hay que tener en cuenta que en algunas BIOS esta facilidad para distinguir entre los distintos discos duros no está presente, o no funciona bien. En los casos en que esto ocurra, tendremos que introducirnos en la BIOS y desactivar los discos duros de los que no queremos que arranque. Así, por ejemplo, en un sistema informático de dos discos duros si queremos arrancar desde el primer disco duro no tenemos que hacer nada pero si queremos arrancar desde el segundo disco duro desactivaremos el primero en la BIOS.

Para desactivar los discos duros, hay que entrar en la primera opción de la BIOS y poner None, not instaled, o algo parecido en el tipo de disco duro que queremos desactivar. Esto no quiere decir que dichos discos duros no se usarán durante el funcionamiento normal de la máquina, sino que no se usarán en el proceso de arranque.



Pero con esto conseguimos indicar al sistema informático que disco duro quiero utilizar para el arranque del sistema... pero resulta que en un solo disco duro puedo tener instalado más de un sistema operativo.

¿Cómo se le indica al sistema que quiero arrancar con Windows XP, o con Linux, o con Beos si todos estos SO están instalados en el mismo disco duro?



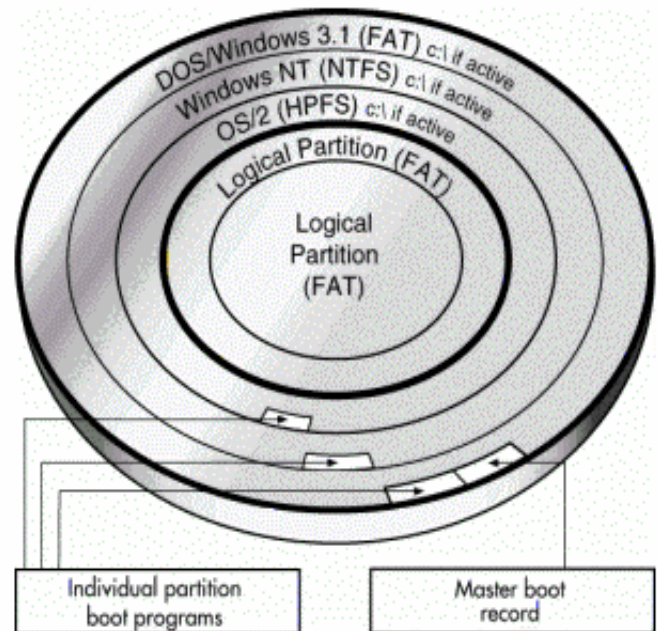
Para entender esto tenemos que comprender bien como está organizado un disco duro.

ORGANIZACIÓN LOGICA DEL DISCO DURO.

Vamos a ver cómo se organiza un disco duro a alto nivel. En el módulo Fundamentos de Hardware tema 2 ya hemos visto cómo se organiza a bajo nivel un disco duro y hemos visto como un disco duro se divide en particiones, los conceptos allí explicados nos serán útiles para entender este tema.

Las particiones son divisiones lógicas efectuadas en un disco duro. Responden a una necesidad muy importante en informática: compartir un mismo disco duro para varios sistemas operativos. Cada partición tiene la estructura lógica correspondiente a su sistema operativo. Una partición Windows 98 contiene sector de arranque, FAT, directorio raíz y área de datos, una partición de Windows en NTFS tiene su sector de arranque y MFT, etc. Los datos de una partición no se mezclan con los de otra.

En un disco duro podemos tener hasta 4 particiones como máximo. De las 4, solo una puede estar definida como activa al mismo tiempo. Esta partición activa será la que cargue el sistema operativo cuando iniciamos el sistema informático.



El primer sector de cada una de estas particiones se conoce como sector de arranque, y en dicho sector (512 bytes) se almacena un programa especial que es el encargado de arrancar el sistema operativo de dicha partición.

En el primer sector del disco duro no se sitúa un sector de arranque, en su lugar se sitúa una tabla de particiones (Master Boot Record o MBR). Esta tabla de particiones incluye una tabla donde definimos las 4 particiones que pueden estar presentes en nuestro disco duro y su tamaño y un pequeño programa que permite localizar la partición activa, leer su sector de arranque y usarlo para arrancar nuestro sistema informático.

Este **MBR** (Master Boot Record) está situado en el primer sector del disco duro, de modo que su tamaño es de 512 bytes. En esta capacidad se almacena lo siguiente por cada MBR:



Dirección.	Contenido.	Tipo.
+000h	Programa MBR.	445 Bytes.
+1BEh	1º entrada de la tabla de particiones	16 Bytes
+1CEh	2º entrada de la tabla de particiones	16 Bytes
+1DEh	3º entrada de la tabla de particiones	16 Bytes
+1EEh	4º entrada de la tabla de particiones	16 Bytes
+1FEh	Identificación (AA55h)	2 Bytes

*Contenido del Master Boot Record o MBR.*

*Longitud = 200h = 512 Bytes.*

*El código AA55h marca este sector como ejecutable.*

Vemos como existe un programa al principio conocido como programa MBR que ocupa 445 Bytes.

Un programa MBR estándar leerá la tabla de particiones y escogerá de cuál de esas particiones va a arrancar el sistema operativo. No lo hará como podría parecer lógico de la primera partición, sino de la partición primaria que está marcada como activa. El MBR lee el primer sector de esa partición, y le cede el control de la CPU a ese programa (Boot Sector o Sector de Arranque).

Hay que indicar que no existe un programa MBR estándar. En realidad, el código que se encuentra aquí, puede ser muy variado, aunque normalmente todos son compatibles. Podemos instalar programas MBR conocidos como gestores de arranque que amplían las posibilidades del gestor de arranque MBR instalado por defecto.

Hay que prestar atención a lo que se ha dicho. Si se arranca desde un disco duro, se lee el primer sector (MBR) y este a su vez, lee un segundo sector (Boot Sector). Vemos también como existen 4 entradas para almacenar hasta 4 particiones, de aquí viene el límite de 4 particiones para un disco duro.

También vemos como por cada partición se almacena su tipo con 16 bytes. En estos 16 bytes se almacena lo siguiente:

Dirección.	Contenido.	Tipo.
+00h	Estado de la partición: 00h – Inactiva 80h – arranque (activa)	1 Byte
+01h	Cabeza de lectura / escritura donde comienza la partición.	1 Byte
+02h	Sector y cilindro donde comienza la partición.	2 Bytes
+04h	Tipo de partición: 00h – Libre 01h – DOS con la vieja FAT de 12 bits. 02h – XENIX 03h – XENIX 04h – DOS con FAT 16 05h – Partición extendida. 06h – Partición DOS > 32 Megas. 0Bh – Windows FAT32 0Ch – Windows FAT 32 LBA 0Eh – VFAT 16h – Hidden FAT 16 (Oculta) 63h – Unix 65h – Novell Netware Etc.....	1 Byte
+05h	Cabeza de lectura / escritura donde termina la partición.	1 Byte
+06h	Sector y cilindro donde termina la partición.	2 Bytes
+08h	Dirección del primer sector de la partición. (Sector de arranque).	4 Bytes
+0Ch	Número de sectores en esta partición.	4 Bytes

*Contenido de cada una de las 4 entradas de la tabla de particiones.  
Longitud = 10h = 16 Bytes.*

Vemos como el 1º campo se usa para indicar si esta partición es la activa o no.

El 2º y 3º campo se usa para indicar el cilindro, sector y cabeza donde comienza la partición.

El 4º campo se usa para almacenar el tipo de la partición, aquí se indica que sistema operativo está instalado en la partición, si dicha partición esta oculta o no, etc.

El 5º y 6º campo se usa para indicar el cilindro, sector y cabeza donde termina la partición.

El 7º campo indica la dirección del primer sector de la partición (el sector de arranque) para que el POST pueda pasarle el control. Este sector siempre es el 1º sector de la partición, pero aquí indicamos su valor director (nº de sector) y no la combinación cilindro, sector y cabeza. Esto se hace para que el acceso al sector de arranque sea más rápido, y para evitar posibles errores en la carga del sistema.

El 8º campo se usa para almacenar el número total de sectores que existen en la partición. Es un campo que se usa para comprobar que los datos de la partición son correctos.

Las particiones de un disco duro pueden ser de dos tipos:

- Primarias
- Extendidas

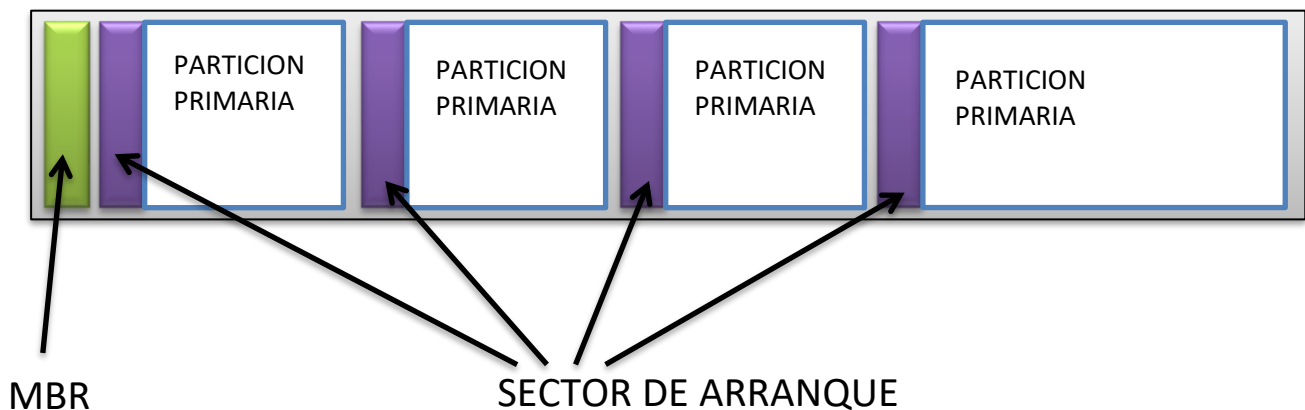
Como ya vimos anteriormente, en un disco duro puede existir un máximo de 4 particiones, sin embargo, sólo una de estas particiones puede ser extendida. (Si seguimos las recomendaciones del estándar usado para organizar lógicamente los discos duros).

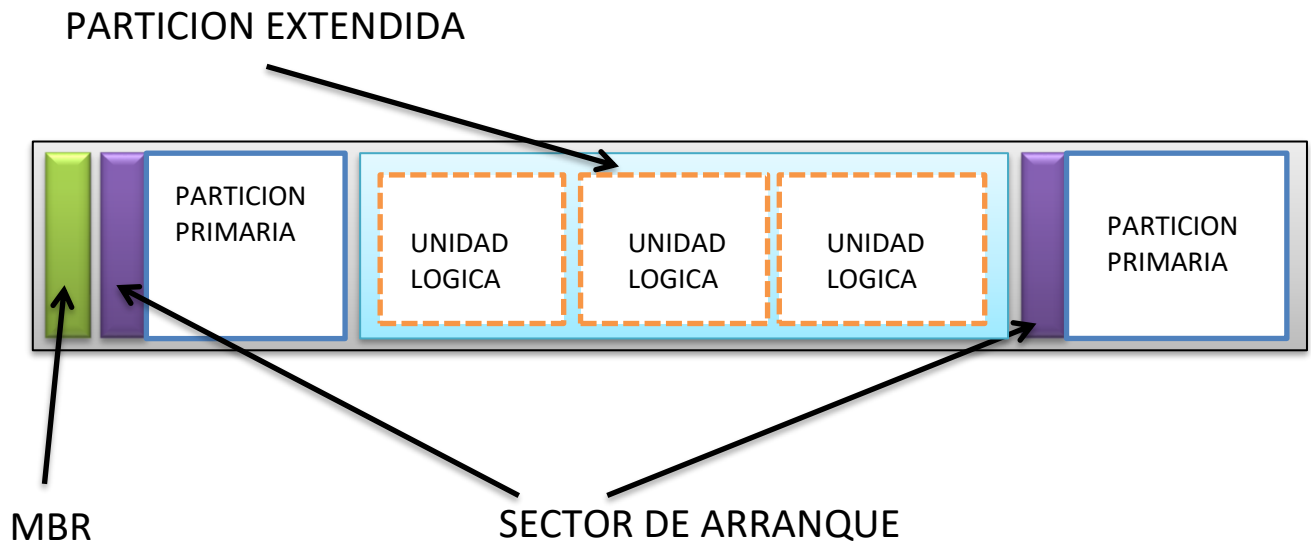
Cada partición primaria forma su propio volumen de datos (la letra en Windows, para entendernos) y tiene su propio sector de arranque. Son las particiones normales.

Una partición extendida, sin embargo, no forma ningún volumen, ni tiene un sector de arranque como tal. Una partición extendida en realidad es un contenedor de unidades lógicas. Se pueden crear tantas unidades lógicas en una partición extendida como se deseen. A términos prácticos, estas unidades lógicas se comportan como particiones primarias.

Cada unidad lógica que se crea dentro de una unidad extendida forma su propio volumen, aunque no tiene un sector de arranque real, sino que usa su sector de arranque para controlar su tamaño entre otras cosas.

De esta manera, si dividimos un disco duro en una partición primaria (un volumen) y una partición extendida (donde creamos 10 unidades lógicas, cada una con su propio volumen) formaremos un total de 11 volúmenes (11 letras de unidad) pero solo tendremos un sector de arranque usable como tal, el de la partición primaria.

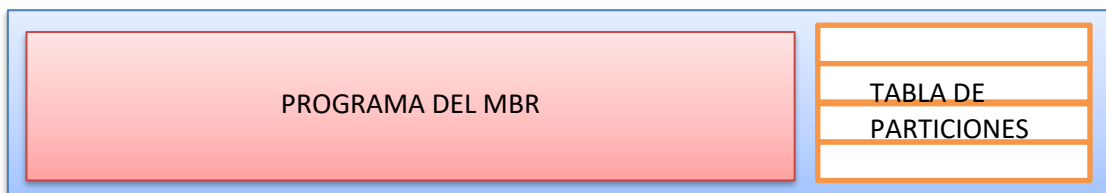




Solo el sector de arranque de una partición primaria es válido para arrancar el sistema operativo. El sector de arranque de la partición extendida solo contiene información sobre las unidades lógicas que se encuentran dentro de ella (tamaños, comienzos y finales, etc.).

La tabla del MBR identifica la localización y tamaño de la partición extendida, pero no contiene información sobre las unidades lógicas creadas dentro de esta partición extendida. Ninguna de estas unidades lógicas puede ser marcadas como activas, por lo que es posible que instalemos un sistema operativo en alguna de estas particiones lógicas, pero nunca podrá ser cargado **directamente**, ya que no podemos marcar esa partición como activa, y por lo tanto no podemos indicar que sea el volumen de arranque. (Para poder instalar sistemas operativos en estas unidades lógicas, tendremos que usar un programa conocido como gestor de arranque que veremos posteriormente, estos gestores de arranque suelen guardar los programas usados para cargar los sistemas operativos siempre en la partición activa del disco duro).

Hemos visto como el MBR se divide en dos partes bien diferenciadas, el programa MBR que ocupa la mayor parte del MBR y la tabla de particiones vista anteriormente.

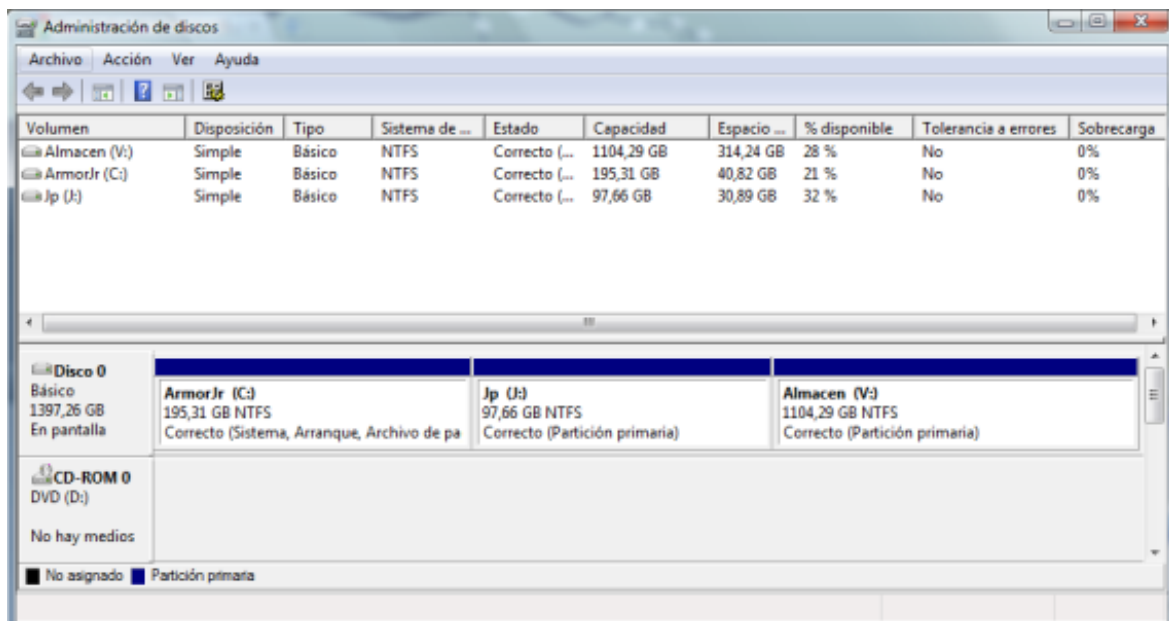


Existen diversos programas que nos permiten gestionar y retocar estos componentes del MBR.

Así, en Windows tenemos el comando FIXMBR que reinstala el programa del MBR, aunque este comando solo podemos usarlo desde la consola de recuperación. (Ya veremos cómo acceder a dicha consola en el siguiente tema).

La tabla de particiones, puede ser gestionada por diversos programas que se incluyen en los sistemas operativos. En sistemas Windows 9x, la utilidad encargada de esto es el FDISK. En la familia Windows más moderna (XP, Vista, 7, 2003, 2008) es la consola del administrador de discos (diskmgmt.msc). Esta consola es la incluida oficialmente por la propia Microsoft, y existen multitud de programas de terceras compañías que permiten retocar esta tabla de particiones. (No es recomendable el uso de dichas herramientas pues pueden estropear la tabla, y suelen dar problemas a la larga). En la familia Windows 2008, Vista, Windows 7 encontramos también una herramienta de línea de comandos que permite gestionar las particiones, diskpart.exe.

Linux por su parte incluye varios programas de este tipo, como pueden ser fdisk, qtparted, parted, etc.



Los Windows modernos (a partir de ahora les llamaremos Windows de la familia NT, o Windows NT) permiten indicar que letra de unidad se le asignará a cada partición, sin embargo DOS y Windows 95/98 asignaban estas letras por defecto. Primero, la C: es asignada a la partición primaria del primer disco donde se encuentre un sistema de ficheros FAT. Entonces la siguiente letra es asignada a la partición primaria con FAT del segundo disco, etc.

Una vez acabadas con las particiones primarias de cada disco, se empiezan a asignar letras a las unidades lógicas del primer disco, luego a las unidades lógicas del segundo disco, etc. Una vez acabado con las unidades lógicas se continúa con el resto de particiones primarias que queden.

Veamos un ejemplo sobre esto. Un usuario tiene un único disco duro dividido en una partición primaria (C:) y un volumen lógico en una partición extendida (D:). Ahora este mismo usuario compra un segundo disco duro y lo instala, creando otra partición primaria y otra partición extendida, conteniendo otro volumen lógico.

Pues bien, después de encender el ordenador, la partición primaria del segundo disco se llama (D:). El volumen lógico del primer disco, que antes se llamaba D pasa a llamarse (E:) y por fin, el volumen lógico del segundo disco recibe el nombre de (F:). Este tipo de cambios era muy peligroso, ya que al cambiar los nombres de las unidades es muy probable que muchos programas dejen de funcionar. Indicar que puesto que las unidades de CD reciben el nombre las ultimas, si este usuario instalase ahora un lector de CD, recibiría el nombre de (G:).

Este problema ocasionado por los sistemas operativos antiguos de Microsoft DOS y Windows 98 no está presente en los sistemas operativos moderno de Microsoft. Así, por ejemplo, Windows XP asigna a cada unidad una letra según lo que hemos visto anteriormente, pero si se encuentra con una unidad que ya ha recibido nombre, no lo cambia.

Linux por su parte no presenta problemas de este tipo, ya que no asigna letras a los volúmenes, en su lugar tenemos que montar cada volumen en un directorio de nuestro árbol de directorios, por lo que no le afectan los problemas de nominación de volúmenes.

Hay que tener mucho cuidado al trabajar con las particiones. La tabla MBR es una tabla muy sensible a cualquier tipo de cambios. Una mala elección de cualquiera de sus campos, puede llevar a la inutilización total del disco duro. Además, dada la facilidad para “trastear” con la tabla de particiones, muchos programas utilizan configuraciones extrañas que son desconocidas para otros programas, lo que puede llevar a perder particiones o a cambiar su tamaño de modo incorrecto.

Desde línea de comandos podemos gestionar particiones con el comando DISKPART. Vemos aquí uno ejemplo de uso:

```
DISKPART> list disk

  Núm Disco  Estado      Tamaño  Disp   Din  Gpt
  -----  -
  Disco 0    En línea    60 GB   0 B
* Disco 1    En línea    2048 MB 2046 MB

DISKPART> select disk 1

El disco 1 es ahora el disco seleccionado.

DISKPART> create partition primary size=300

DiskPart ha creado satisfactoriamente la partición especificada.
```

Una vez creada la partición, tenemos que formatearla y asignarle una letra:

```
DISKPART> list partition
  Núm Partición  Tipo                Tamaño  Desplazamiento
-----
* Partición 1    Principal                300 MB    64 KB
DISKPART> select partition 1
La partición 1 es ahora la partición seleccionada.
DISKPART> format
  100 por ciento completado
DiskPart formateó el volumen correctamente.
DISKPART> assign letter=f
DiskPart asignó correctamente una letra de unidad o punto de montaje.
```

Vemos como las acciones de crear partición, formatear partición y asignar letra a partición se separan cada una en su propio comando.

Para ver la ayuda de todos los comandos posibles que podemos ejecutar en DISKPART podemos ejecutar el comando HELP. DISKPART no solo permite trabajar con discos duros básicos, sino también con discos duros dinámicos (que utilizan LVM (gestor lógico de volúmenes) que es un concepto que veremos en segundo).

GPT.

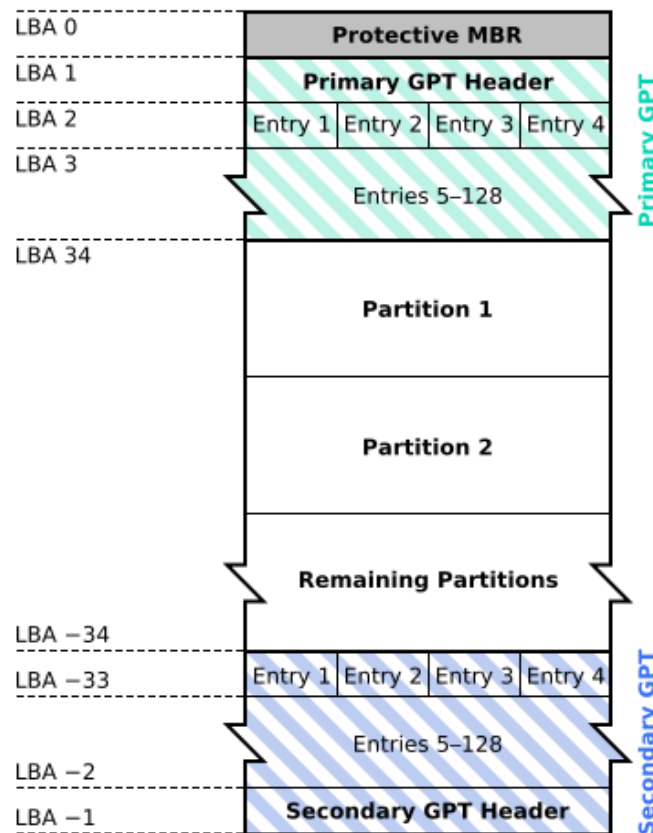
Hemos visto en el punto anterior como funciona un disco duro con una tabla de particiones MBR, que es la opción más habitual con la que nos vamos a encontrar. Sin embargo, desde hace un tiempo se está substituyendo nuestras antiguas BIOS por un sistema más moderno conocido como EFI (Extensive Firmware Interface).

Este sistema, totalmente incompatible con BIOS, permite que en el disco duro nos olvidemos si queremos de la MBR y utilicemos un sistema mucho más potente, conocido como GPT (GUID Partition Table, siendo GUID acrónimo de Globally Unique Identifiers).

GPT usa un moderno modo de direccionamiento lógico (LBA, logical block addressing) en lugar del modelo cilindro-cabeza-sector (CHS) usado con el MBR. La información de MBR heredado está almacenada en el LBA 0 o bloque lógico 0, la cabecera GPT está en el LBA 1, y la tabla de particiones en sí en los bloques sucesivos. En los sistemas operativos Windows de 64-bits, 16.384 bytes, o lo que es lo mismo, 32 sectores, están reservados para la GPT, dejando el bloque LBA 34 como el primer sector usable del disco.

GPT proporciona asimismo redundancia. La cabecera GPT y la tabla de particiones están escritas tanto al principio como al final del disco.

### GUID Partition Table Scheme





Vemos como al principio del disco se guarda un sector conocido como protective MBR. El propósito de este sector es permitir que programas y sistemas que están preparados solo para trabajar con MBR y no con GPT puedan ver el disco como válido.

Este MBR “falso” está configurado con una sola partición que ocupa todo el disco, y es totalmente obviado por EFI, por lo que no se utiliza nunca. Sin embargo, si un programa o sistema operativo antiguo intenta usar este disco, creará que el disco duro es un MBR normal de 1 sola partición, y cuando intente acceder al disco duro, se dará cuenta que la información almacenada en el no coincide con lo que el espera, y mostrará un mensaje indicando que la estructura del disco duro esta corrompida, que no encuentra el sistema, o algo parecido.

Es este el gran fallo de GPT, que solo es compatible con los nuevos sistemas operativos y los nuevos programas. Por poner un ejemplo, si instalamos Windows 7 64 bits configurando el disco como GPT, si luego queremos instalar un sistema operativo anterior en ese mismo disco duro, el propio sistema nos indicará en la instalación que no puede trabajar con el disco duro ya que esta corrompido, y no podremos instalar el sistema.

Por ejemplo, todas las versiones de Windows 7 pueden leer discos GPT, pero solo las versiones de 64 bits de Windows 7 pueden arrancar desde un disco duro GPT.

La cabecera de la tabla de particiones (Primary GPT Header) define los bloques de disco que pueden ser utilizados por el usuario (bloques usables). También define el número y tamaño de las entradas de partición que conforman la tabla de particiones. En Windows hay 128 entradas de partición reservadas, cada una de 128 bytes de longitud. Así, se pueden crear hasta 128 particiones si usamos un sistema tipo Windows.

La cabecera contiene el GUID del disco (Globally Unique Identifier, Identificador Global Único). Registra su propio tamaño y localización (siempre LBA 1), y el tamaño y la localización de la cabecera y tabla de la GPT secundarias (siempre en el último sector del disco). Es importante que también contiene una suma de comprobación CRC32 para sí mismo y para la tabla de partición, que se verifica por los procesos EFI durante el arranque. Además, todo GPT está configurado para usar Unicode.

Para entender mejor por qué se crea GPT, veamos los principales problemas que presentan MBR y las ventajas que aporta GPT.

---

#### PROBLEMAS CON EL MBR.

1. En MBR sólo pueden ser definidas 4 particiones primarias o 3 primarias + 1 partición extendida (con un número arbitrario de particiones lógicas dentro de la partición extendida).
2. En MBR dentro de la partición extendida, los metadatos de las particiones lógicas se almacenan en una estructura de lista enlazada. Si un enlace se pierde, todas las particiones lógicas existentes, después de los metadatos, se pierden.
3. MBR sólo admite 1 byte para códigos de tipo de partición, lo que conlleva muchas colisiones.
4. MBR almacena la información del sector de la partición con valores LBA de 32 bits. Esta longitud de LBA junto con los 512 byte del tamaño del sector (más comúnmente utilizados) limita el tamaño máximo manejable del disco hasta 2 TB.

---

#### VENTAJAS DE GPT.

1. Utiliza GUID (UUID) para identificar los tipos de particiones. Sin colisiones.
2. Proporciona un GUID único de disco y un GUID único de partición para cada partición. Un buen sistema de archivos independiente referenciando a las particiones y discos.
3. Número arbitrario de particiones (depende del espacio asignado por la tabla de particiones). No hay necesidad de particiones extendidas y lógicas. Por defecto, la tabla GPT contiene espacio para la definición de 128 particiones. Sin embargo, si el usuario desea definir más particiones, se puede asignar más espacio (de momento solo en Linux).
4. Utiliza 64-bit LBA para almacenar números del Sector - tamaño máximo del disco manejable es de 2 Zeta Bytes.
5. 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 están dañados.
6. Código de reparación de errores CRC32 para detectar errores y daños de la cabecera y en la tabla de particiones.

## ARRANQUE DE WINDOWS XP/2000/2003

Veamos ahora el arranque de un sistema operativo Windows XP, 2000 o 2003.

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 **NTLDR** (LoaDeR de NT)
5. NTLDR ajusta el procesador para trabajar a 32 bits o 64 bits.
6. NTLDR lee el fichero **BOOT.INI** 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. NTLDR carga **NTDETECT.COM**
9. NTDETECT.COM genera la lista de hardware. Devuelve el control a NTLDR
10. NTLDR carga **NTOSKRNL.EXE** (Núcleo del sistema operativo o Kernel).
11. NTOSKRNL.EXE lee el **registro** de Windows, y procede a ir cargando el sistema completo.

NTOSKRNL.EXE como indica es en gran parte el kernel o núcleo del sistema operativo, y es un programa de gran tamaño que se encuentra normalmente en nuestro directorio Windows. Sin embargo, tanto ntlldr, como boot.ini o ntddetect.com son programas pequeños.

Esto permite que podemos situar dichos ficheros en un disquete, llavero USB, etc, con lo que tendríamos un volumen de INICIO, lo que nos permitiría iniciar el sistema aunque el disco duro haya sufrido algún problema. Sin embargo, no se puede confundir este “disco de inicio” con un “disco de arranque”. Cuando llegue el momento de cargar NTOSKRNL.EXE si no se encuentra, el sistema se detendrá y no arrancará, y por el tamaño de dicho fichero y de todos los que necesita para trabajar, es imposible copiarlo en un volumen si no es de gran tamaño.

Es importante conocer esta secuencia, para comprender los distintos errores que se pueden cometer y con los que nos podemos encontrar. Por ejemplo, si recibimos el mensaje “falta ntlldr” al intentar arrancar, está claro que se ha producido un error en el punto 4, lo que nos indicaría que se ha leído el MBR, el sector de arranque, y no se ha encontrado en el raíz de nuestro volumen el fichero ntlldr, bien porque lo hayan borrado o por que se haya borrado todo el volumen.

Sin embargo, un mensaje “falta ntoskrnl.exe” nos indicaría que si existe un fichero ntlldr, pero que en nuestro directorio de Windows no se ha encontrado un fichero NTOSKRNL.EXE.



## ARRANQUE DE WINDOWS 7/VISTA/2008/8/10

La secuencia de arranque de Windows cambió a partir de XP. La principal diferencia estriba en que se ha cambiado el gestor de arranque, ya no se usa el ntlldr sino que se usa el Windows Boot Manager (bootmgr).

Mientras que el gestor ntlldr 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.

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.

Windows dispone de un comando para configurar esta base de datos BCD, el **bcdedit.exe**, pero es realmente engorroso de usar. Es mejor usar una utilidad grafica de una 3rd party (tercera compañía, una compañía distinta a la que realiza el sistema) como puede ser EasyBCD que permite configurar muchas más opciones que el bcdedit.exe y de forma mucho más fácil.

## ARRANQUE DE WINDOWS 8/10

Aunque el arranque de Windows actual es muy similar al de Windows 7 incorpora varias novedades, muchas de ellas basadas en el uso de UEFI en lugar de BIOS. Una de las más importantes es la del Secure Boot.

### SECURE BOOT.

Los ordenadores cuando encontraban el sector de arranque del SO que querían cargar, se limitaban a ejecutar dicho código, sin comprobar de ningún modo qué es lo que se está ejecutando.

Sin embargo, si contamos en el sistema con UEFI en lugar de BIOS y esta activada una característica conocida como Secure Boot, el firmware del sistema comprueba la firma digital del sector de arranque, para comprobar si es de un sistema reconocido, y si se ha producido algún tipo de modificación sobre el mismo. Para permitir el arranque del sistema operativo, se deben dar una de las siguientes situaciones:

- El código de carga fue firmado utilizando un certificado “de confianza”. Por ejemplo un certificado de Microsoft.
- El usuario ha aprobado la firma digital del código de carga. UEFI debería permitir al usuario realizar esta acción. (No siempre ocurre).
- El usuario deshabilita Secure Boot en la configuración de UEFI.
- El usuario deshabilita totalmente UEFI, y en su lugar utiliza BIOS.

### TRUSTED BOOT.

Una vez que secure boot ha terminado su cometido, el código de carga (bootloader) verifica el firmado del kernel de Windows 8 antes de cargarlo. A su vez, el kernel de W8 verifica todos los componentes de Windows que se van cargando, incluyendo los drivers de dispositivo de la propia Microsoft que se cargan en el arranque. Si un fichero ha sido modificado, el bootloader detecta el problema y se niega a seguir cargando el componente. Windows 8 intenta reparar el componente corrupto automáticamente.

### FAST STARTUP

Windows Fast Startup (Inicio rápido) es la opción por defecto a utilizar en Windows 8, Windows Server 2016 y Windows 10 siempre que se utilice UEFI.

En un sistema Windows en cada momento se encuentran ejecutándose dos sesiones en realidad, la del usuario actual y la del kernel del sistema. Cuando en Windows 7 se apaga el sistema, se cierran ambas sesiones y hay que volver a cargarlas desde cero cuando el sistema se inicia.

Windows 10 cierra totalmente la sesión del usuario y la vuelve a cargar en cada inicio, sin embargo, la sesión del kernel la hiberna, leyendo todo su estado en la RAM y grabándolo directamente en el disco duro. Esto permite que cuando el sistema se inicie, no se vea obligado a volver a leer todos los archivos del kernel, sino que directamente recupera el estado desde el disco duro hasta la RAM. Esto permite que se inicie Windows 10 mucho más rápido que Windows 7.

Esta hibernación se realiza solo con la sesión de kernel porque es pequeña y predecible, mientras que no se realiza con la sesión de usuario porque suele ser mucho más grande, y es impredecible (igual ocupa muy poco que muchísimo).

En el caso de que contemos con varios sistemas operativos instalados en la misma máquina, si cambiamos de un sistema operativo al otro se descarta la sesión de kernel grabada en el disco duro y veremos cómo se “reinicia” la máquina dos veces.

Este fast startup nos puede dar problemas en determinados escenarios:

Cuando se apaga un equipo con Fast Startup activado, Windows bloquea el disco duro. No podremos acceder al mismo desde otro sistema operativo. Si se da la circunstancia que intentemos usar ese disco duro en otra máquina nos podemos encontrar con casos en los que se corrompe la información almacenada.

Dependiendo de nuestro sistema, puede que no seamos capaces de acceder a la configuración de nuestro BIOS/UEFI settings si tenemos Fast Startup activado y hemos hibernado nuestra sesión de kernel. Cuando hibernamos un equipo, este no entra en modo de apagado completo y luego no experimenta un encendido completo (se suele hablar de encendido caliente y encendido frío)

Si queremos apagar totalmente nuestro Windows 10 haciendo que no se hiberne la sesión y que realice un arranque completo la próxima vez que se encienda el equipo, hay que ejecutar el siguiente comando en un interfaz de línea de comandos con poderes de administrador:

```
shutdown /s /t 0
```

También podemos desactivar totalmente Fast Startup con la instrucción

```
powercfg /hibernate off
```

y volver a activarla cuando queramos con

```
powercfg /hibernate on
```

## ARRANQUE DE LINUX. GRUB.

Linux no cuenta con un gestor de arranque propio, sino que permite usar cualquier gestor de arranque que deseemos. El que se suele incluir actualmente en todas las versiones de Linux es el GRUB.

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.

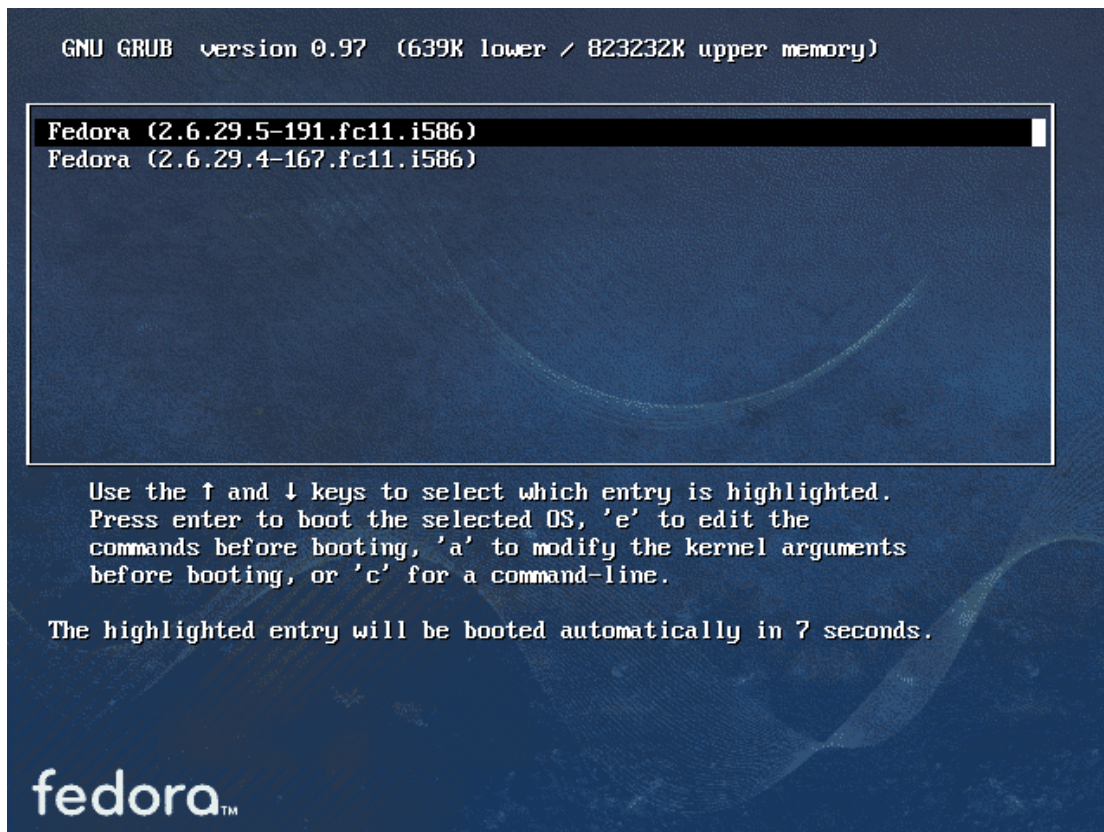
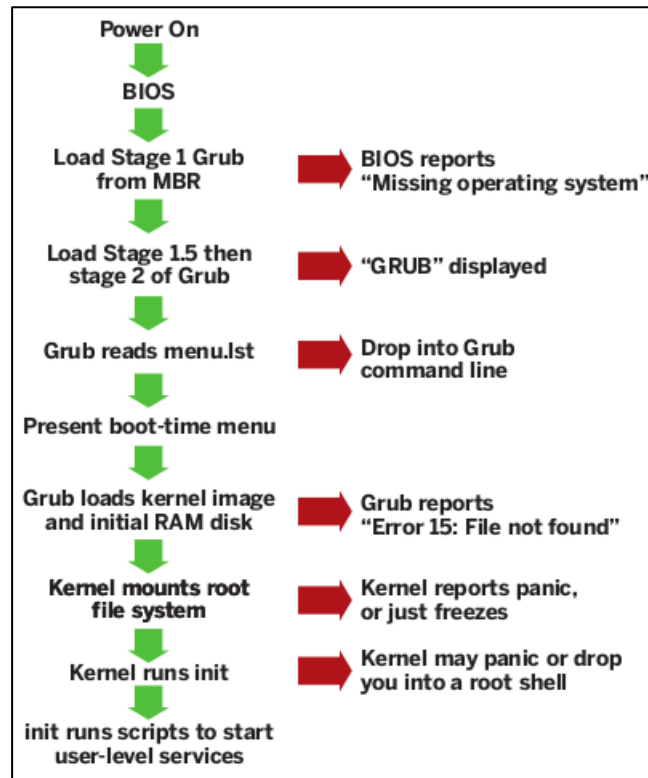
En la actualidad nos podemos encontrar con GRUB en sus versiones 1 y 2, que son algo distintas.

El proceso de inicio de GRUB 1 es el siguiente:

1. La BIOS busca un dispositivo de inicio (como el disco duro) y pasa el control al registro maestro de inicio (Máster Boot Record, MBR, los primeros 512 bytes del disco duro).
2. El MBR contiene la fase 1 de GRUB. Como el MBR es pequeño (512 bytes), la fase 1 sólo se encarga de buscar y cargar la siguiente fase del GRUB (ubicado físicamente en cualquier parte del disco duro). La fase 1 puede cargar ya sea la fase 1.5 o directamente la 2
3. GRUB fase 1.5 está ubicada en los siguientes 30 kilobytes del disco duro. La fase 1.5 carga la fase 2. Esta fase es optativa y normalmente no se usa.
4. GRUB fase 2 (cargada por las fases 1 ó 1.5) recibe el control, y presenta al usuario el menú de inicio de GRUB. Este menú se configura mediante un fichero de texto con nombre menu.lst.
5. GRUB carga el kernel (núcleo) seleccionado por el usuario en la memoria y le pasa el control para que cargue el resto del sistema operativo.

GRUB 2 sustituye el fichero menu.lst (que editamos manualmente) por un proceso modular, de modo que automáticamente se añaden los sistemas operativos y las opciones de los mismos. Ya veremos en profundidad estos gestores en los temas dedicados a GNU/Linux.

GRUB no es en realidad un gestor de arranque para Linux, sino un gestor de arranque para cualquier sistema operativo. De hecho, GRUB es perfectamente capaz de arrancar cualquier sistema operativo de la familia Windows sin ningún tipo de problemas. Vemos aquí una lista cronológica indicando en que momento aparece cada sistema operativo.





## RECUPERACIÓN DE ERRORES EN EL ARRANQUE.

El proceso de arranque es un concepto al que el administrador de sistemas debe prestarle mucha atención, dado que el más mínimo problema que se origine en dicho proceso, hará imposible que el sistema operativo arranque, y por lo tanto dejara inservible el sistema informático.

Las zonas que hay que vigilar y conocer cómo recuperar si es necesario, son el MBR, el sector de arranque de la partición primaria activa y el programa gestor de arranque que este situado en dichas zonas.

¿Pero, que errores se pueden producir en el arranque?

En primer lugar debemos hablar de los fallos de hardware. Al usar un disco duro siempre existe la posibilidad de que se corrompan clústeres del mismo. Normalmente estos errores no suelen tener demasiada importancia, pero si se da la casualidad de que se corrompe el primer clúster del disco duro, que es donde se sitúa el sector del MBR y el primer sector de arranque de la primera partición, nos vamos a encontrar en serios problemas. Normalmente en estos casos lo mejor es cambiar el disco duro completo, e intentar recuperar la información que existía en el disco duro con algún programa de recuperación de datos profesional.

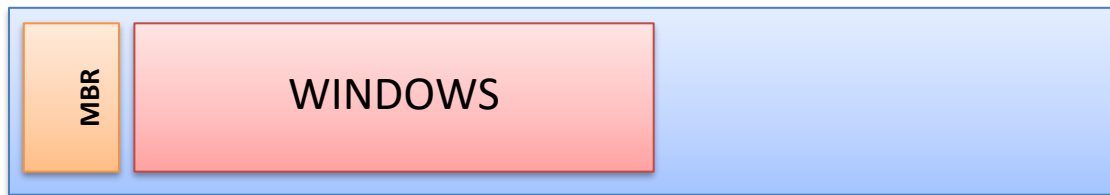
En segundo lugar nos encontramos la acción del malware (virus, gusanos, troyanos, etc.). Estas amenazas pueden borrar el MBR y los sectores de arranque, y antiguamente existían bastantes virus que se dedicaban a realizar estas acciones. Hoy en día, y con la “profesionalización” de los desarrolladores de malware, estas prácticas han quedado relegadas al olvido.

La tercera causa, y la que suele ser culpable en el 99% de los casos, es que directamente el usuario estropee el arranque de un sistema operativo, simplemente instalando un segundo operativo. Veamos con detalle esta situación:

Hemos visto como cada sistema operativo cuenta con su propio programa para instalar en el MBR, su propio programa para instalar en el sector de arranque, y también cuentan con su propio gestor de arranque.

Está claro que si instalamos en un mismo disco duro tres sistemas operativos distintos, cada uno de ellos habrá ido instalando su propio proceso de arranque, pero como solo puede existir un proceso de arranque en un disco duro (sólo existe un MBR) el proceso de arranque que se quede al final será el del ultimo sistema operativo instalado, que machacará el proceso de arranque del sistema operativo anteriormente instalado, y así sucesivamente.

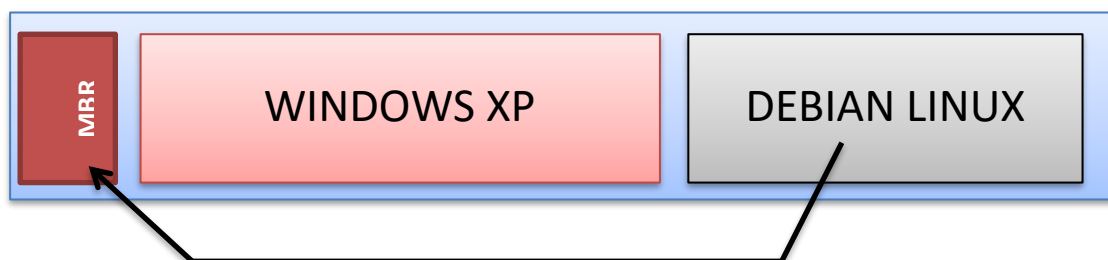
Imaginemos el caso siguiente: En un disco duro tenemos instalado una partición con Windows



En el MBR tendremos instalado evidentemente el gestor de arranque de Windows, y en la partición de Windows tendremos instalado los archivos que necesita el gestor de arranque de Windows para funcionar.

Decidimos instalar en dicho disco duro una distribución de Linux como Debian, para lo cual le creamos una partición y procedemos a instalar dicho sistema operativo:

Durante este proceso de instalación, Debian instalará (si no lo evitamos) en el MBR el gestor de arranque de Debian (en este caso grub), y por lo tanto machacará al gestor de arranque de Windows que estaba anteriormente instalado en el MBR.



La próxima vez que iniciemos la máquina, se cargará el gestor de arranque de grub, no el anterior que teníamos de Windows. ¿Reconocerá el gestor de arranque de grub que en el disco duro existe un Windows y nos permitirá arrancar desde el, aparte de arrancar desde Debian? Pues en este caso sí. En el mundillo de los gestores de arranque, es conveniente recordar siempre estas pequeñas reglas:

- 1) Grub es capaz de arrancar cualquier sistema operativo, por lo que respetará siempre (o al menos lo intentará) cualquier sistema operativo que hubiera en disco duro antes de que se instalara dicho gestor de arranque.
- 2) Los gestores de arranque de Windows nunca respetarán a Linux. De hecho, el gestor de arranque de Windows solo es capaz de arrancar automáticamente a sistemas operativos Windows, siendo muy complicado conseguir arrancar otros sistemas operativos no de Microsoft.

- 3) Los gestores de arranque de Windows respetan a los sistemas operativos Windows pero solo a los anteriores a dicho Windows. Es decir, Windows 8 reconoce y respeta a Windows 7, pero al contrario no, ya que cuando se creó el gestor de arranque de 7 el sistema operativo Windows 8 no existía, y por lo tanto dicho gestor de arranque no lo reconocerá como un SO legítimo, y por lo tanto se negará a arrancarlo de forma automática.

Comprobad cuales de las siguientes instalaciones de sistemas operativos en un mismo disco duro, darían problemas y cuales no:

- a) Instalamos Windows 7, luego Windows 8 y por ultimo Windows 10. ¿Daría problemas? ¿Qué sistemas operativos aparecerían para escoger en el menú de arranque?
- b) Instalamos Linux, luego Windows 10 y por ultimo Windows 7.
- c) Instalamos Windows 10, luego Windows 8 y por ultimo Windows 7.
- d) Instalamos un Windows 10 y luego otro Windows 10.

Cada sistema operativo cuenta con herramientas que permiten reconstruir el programa gestor de arranque en el MBR, y arreglar los sectores de arranque.

---

#### WINDOWS XP.

En el caso de Windows XP hay que iniciar el sistema desde el CD original de instalación de Windows XP. En el proceso de instalación que se ejecutará, hay que llegar hasta el punto en que nos permite ejecutar la “consola de recuperación”. En dicha consola podremos ejecutar desde líneas de comandos las siguientes órdenes:

*FIXMBR*            *Instala el gestor de arranque de XP en el MBR.*  
*FIXBOOT*          *Recupera el sector de arranque de Windows XP.*

También existen órdenes para recuperar la lista de sistemas operativos que aparecen en el menú, pero eso lo veremos en un tema posterior.

---

#### WINDOWS 7.

Igualmente que en el punto anterior, tenemos que iniciar el sistema desde el CD original de instalación de Windows 7. Llegará un momento en que el propio programa de instalación nos dará la opción de realizar una reparación automática del inicio de Windows. Escogemos esta opción y comprobamos si el sistema es capaz de repararse automáticamente. Si comprobamos que dicho automatismo falla (cosa bastante probable) volvemos a iniciar el sistema desde el CD, pero esta vez desde el menú avanzado escogemos la opción de consola de recuperación o línea de comandos. Desde allí podemos ejecutar las siguientes órdenes:

*Bootrec.exe /fixmbr*    *Instala el gestor de arranque de Windows 7 en el MBR.*  
*Bootrec.exe /fixboot*   *Recupera el sector de arranque de Windows 7.*

---

## WINDOWS 8/10/2016

Las instrucciones que hemos visto anteriormente para Windows 7 funcionan exactamente igual en Windows 8.

---

## LINUX.

En este caso iniciamos el sistema desde un CD especial para recuperación del grub, como puede ser por ejemplo el “súper grub disk” o “súper grub2 disk”. También podemos recuperar el sistema arrancando desde un cd de una distribución “live”. Este tema lo dejamos para cuando nos hayamos familiarizado con Linux.

---

## PROBLEMAS DE ARRANQUE CON UEFI Y BIOS.

UEFI (Unified Extensible Firmware Interface) es una interfaz de firmware estándar para PCs, diseñada para reemplazar BIOS (sistema básico de entrada y salida). Es un estándar creado por más de 140 compañías tecnológicas que forman parte del consorcio UEFI, en el que se incluye Microsoft. Se ha diseñado para mejorar la interoperabilidad del software y solucionar las limitaciones del BIOS. Algunas de las ventajas que ofrece el firmware UEFI son:

- Ayudar a proteger el proceso previo al inicio frente a ataques de bootkit.
- Tiempo de inicio y reanudación desde la hibernación más rápidos
- Compatibilidad con unidades de disco duro con particiones de más de 2,2 TB.
- Compatibilidad con controladores de dispositivos de 64 bits.
- Capacidad para usar Secure Boot.

UEFI es el firmware que ahora mismo podemos encontrar en los PC comerciales, es muy extraño encontrar un equipo que no lo soporte. En BIOS de tipo UEFI únicamente podemos instalar los sistemas de 64 bits. Los de 32 bits no se pueden instalar en modo UEFI. La UEFI es un BIOS mucho más amigable que la clásica BIOS con pantalla azul, soporta un entorno gráfico de mayor calidad, multilinguaje, precarga de aplicaciones o gestión de LAN, entre otras muchas opciones.

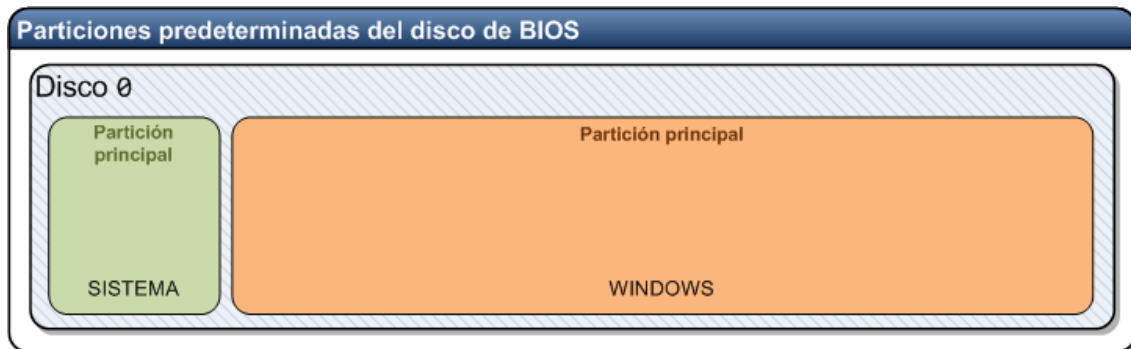
El gran problema que nos podemos encontrar en el arranque, es que actualmente podemos configurar el BIOS bien en modo compatibilidad (legacy) o en modo UEFI puro. Bien, pues un sistema operativo que se haya instalado en modo UEFI no podrá ser cargado por un sistema que se pase a BIOS, y viceversa. Esto hace que muchas veces nos encontremos con sistemas que no arrancan simplemente porque no se ha escogido la versión correcta.

Un disco duro instalado nuevo bajo UEFI tendrá una estructura como la siguiente:



Vemos como se crea de forma predeterminada una partición de sistema Extensible Firmware Interface (partición de sistema EFI), una partición reservada de Microsoft (MSR, Microsoft Reserved Partition) y una partición principal de Windows que es donde se instalará nuestro SO.

En una instalación por defecto BIOS el disco duro quedará de la siguiente forma:

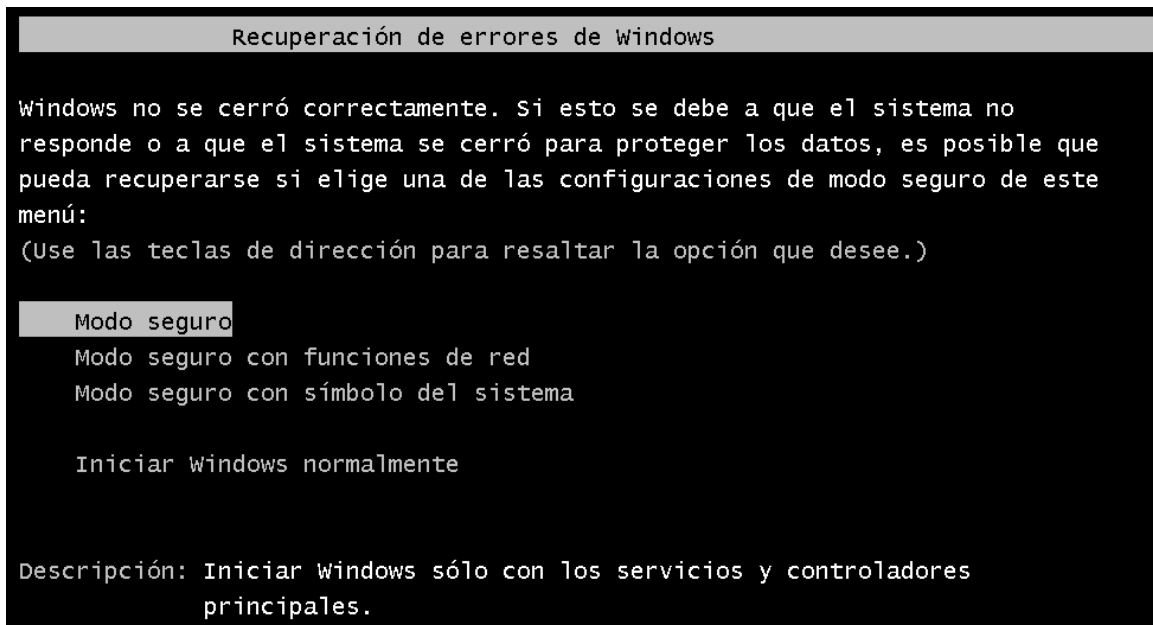


**MODOS DE ARRANQUE A PRUEBA DE FALLOS.**

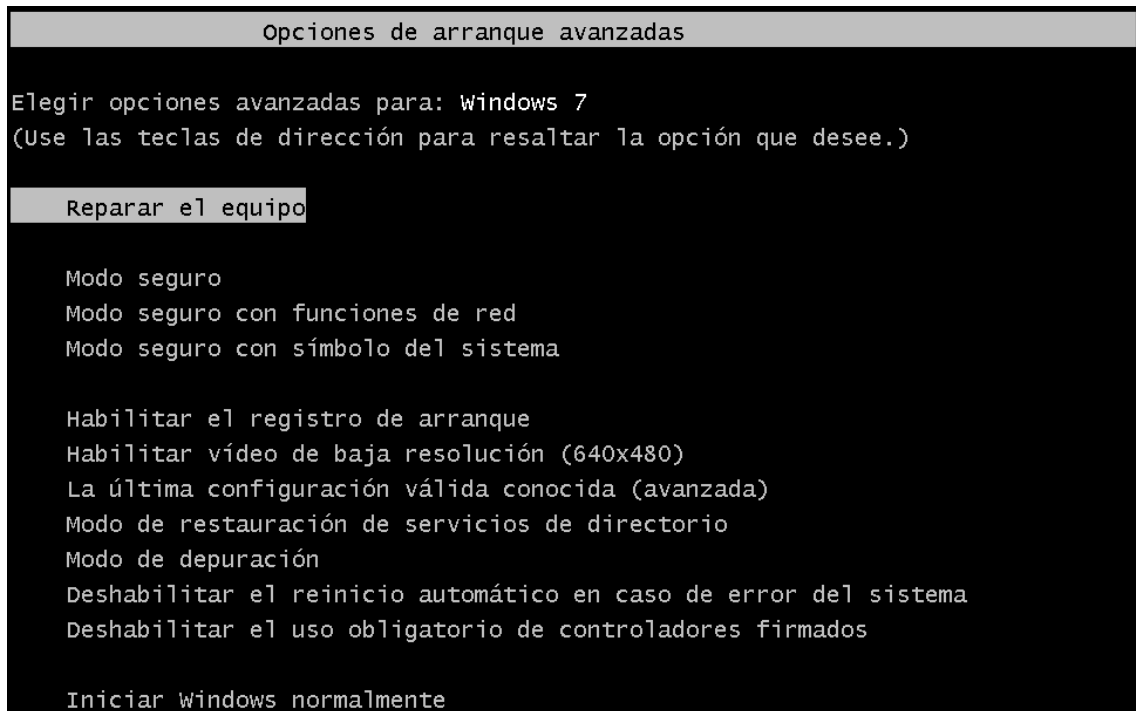
En punto anterior hemos visto cómo podemos solucionar los fallos del arranque más importantes, que conllevan sobrescribir el MBR o bien el sector de arranque. Sin embargo existen otros muchos tipos de errores que se pueden producir en el inicio del sistema operativo, y que no se pueden solucionar con esas técnicas. Errores típicos de este tipo pueden ser la instalación de un driver corrupto, el borrado accidental de un fichero del sistema, etc.

Cuando un sistema no puede iniciarse debido a un error de este tipo, siempre podemos intentar iniciar el sistema operativo en un modo especial conocido como modo a prueba de fallos, donde se cargarán las funciones básicas del sistema, intentando saltarse las partes que pueden estar provocando fallos. Para ingresar en el modo a prueba de fallos en un Windows, basta con pulsar la tecla F8 justo cuando el sistema inicia su carga.

Esta pantalla que vemos aquí por ejemplo, es la que se obtiene si iniciamos Windows 7 después de habernos salido del mismo de una forma descontrolada (apagando el ordenador sin cerrar el sistema, por ejemplo).

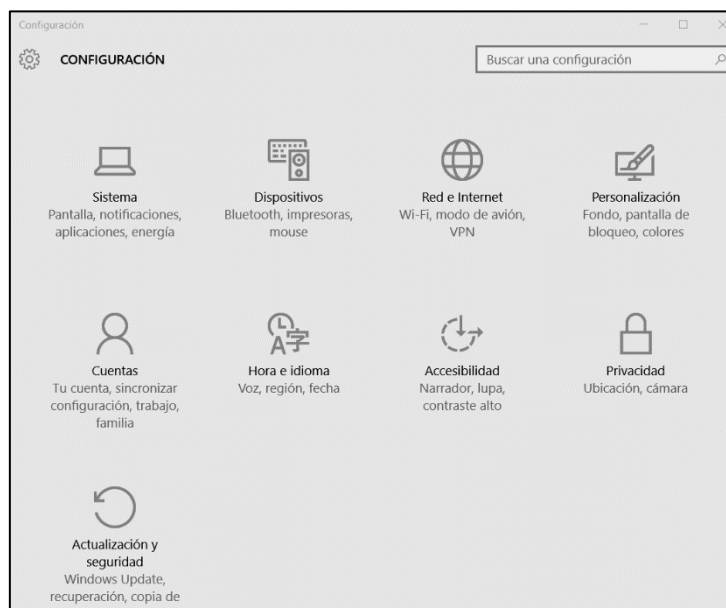


Si pulsamos F8 cuando el sistema se está iniciando, sin embargo, Windows nos presenta la siguiente pantalla:



Vemos como además de los modos seguros, permite arrancar el sistema con otras configuraciones establecidas, como pueden ser con gráficos de baja resolución. Este menú aparece de una forma u otra en todas las versiones de Windows, aunque en Windows 8/10 hay que activarlo antes de poder usarlo desde el panel de control del propio Windows.

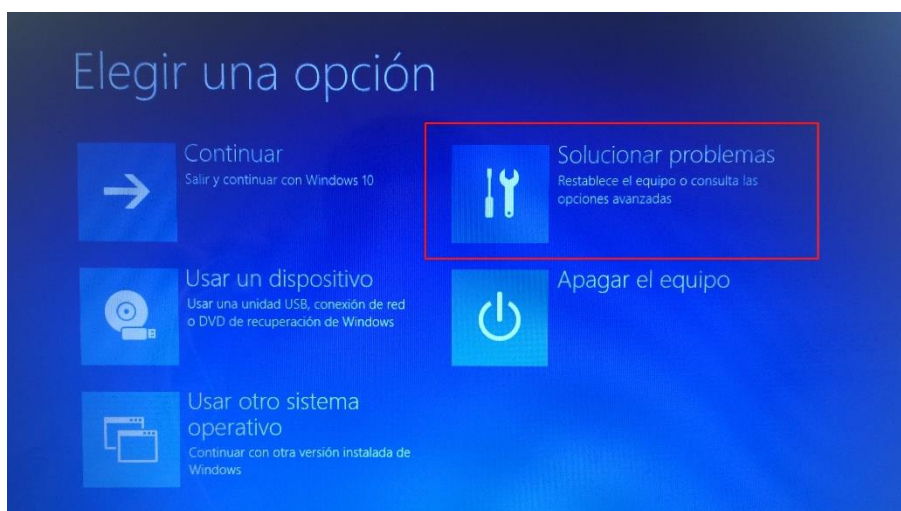
Para hacer esto en Windows 10 hay que ir al menú de configuración de Windows (Windows + I).



En este panel de configuración escogemos Actualización y seguridad. Luego Recuperación e Inicio Avanzado – Reiniciar ahora.



Con lo cual llegaremos a la pantalla de recuperación / inicio avanzado de Windows 10.



La opción apagar el equipo nos permite realizar un apagado total del equipo, para que luego podamos arrancarlo en “frio”, lo que nos permite entrar en el Setup del BIOS, etc. Si no apagamos el equipo así, realmente no se apaga del todo y luego realiza un arranque en “caliente”. Este tipo de arranque no comprueba las pulsaciones del teclado con lo cual no podremos entrar en BIOS.

La opción que nos interesa en este momento es la de solucionar problemas.





Las opciones restaurar y restablecer permiten reinstalar completamente el sistema operativo, bien sin perder archivos o bien realizando un formateo y por lo tanto perdiéndolo todo. A nosotros ahora mismo nos interesa Opciones avanzadas.



Desde aquí podemos realizar muchas cosas (una de ellas es intentar automáticamente reparar el inicio, aunque normalmente no funciona). Las opciones que nos interesan son bien Símbolo del Sistema, para poder acceder a la consola de recuperación y ejecutar los comandos Bootrec como ya vimos anteriormente, o bien Configuración de inicio, que nos mostrará el menú del modo de arranque a prueba de fallos.

En Linux no tenemos un modo seguro como tal, pero podemos pasarle parámetros al kernel indicando como queremos lanzar nuestro Linux, desactivando por ejemplo los gráficos en alta resolución, el multiusuario, los puertos USB, etc. Lo veremos en el tema de Linux.

**EVOLUCIÓN HISTORIA DE LOS SISTEMAS OPERATIVOS.**

A lo largo de la historia, han existido cientos de familias de sistemas operativos, cada una de ellas compuesta por decenas de sistemas operativos distintos. Es una historia que comienza en los años 1960 y que sigue hasta el día de hoy.

Podría parecer que existen pocos sistemas operativos en el mercado, pero si investigamos un poco nos damos cuenta que existen cientos de alternativas posibles. Es más, hoy en día se ha abierto el mercado de los sistemas operativos para móviles y tablets, donde existen decenas de sistemas operativos nuevos que se van presentado cada año.



[Cronología de los sistemas operativos](#)

[Linux distros time line](#)

[From Windows 1 To Windows 10: 30 Years of OS History](#)

[Mac OS Evolution](#)

[Desktop enviroments for Linux](#)