Unidad 2 - Software base de un sistema informático.

1. Introducción a los sistemas operativos.

El ordenador es un sistema programable formado por un conjunto de elementos hardware que necesitan instrucciones que le indiquen cómo utilizar los recursos. El conjunto de instrucciones o programas es lo que conocemos como soporte lógico o software. Un ordenador, sin software que lo programe, es básicamente un bloque de metal inútil, pero con el software puede almacenar, procesar y obtener información, editar textos, controlar el entorno, etc.

1.1. Concepto.

Sin duda alguna, la utilización de los recursos mediante programas es muy complicada, puesto que cada dispositivo es diferente y con gran cantidad de características a controlar. Por ello, una de las primeras acciones a llevar a cabo es el diseño y codificación del software que nos facilite el manejo de estos recursos, evitando, en lo posible, que debamos poseer profundos conocimientos del hardware, cediéndole esta tarea a un reducido número de profesionales que serán los que construyan dicho software. Una vez realizado este esfuerzo de diseño, cabe pensar por que no se completa un poco más con el fin de dotar a los usuarios de unas cuantas funciones adicionales, que no sólo faciliten el uso de estos recursos, sino que además los potencien lo más posible. Pues bien, este software así diseñado, cuya finalidad es gestionar adecuadamente los recursos para que realicen el trabajo que se les ha encomendado, y que, además, potencien las funciones de los mismos, es lo que denominaremos sistema operativo.

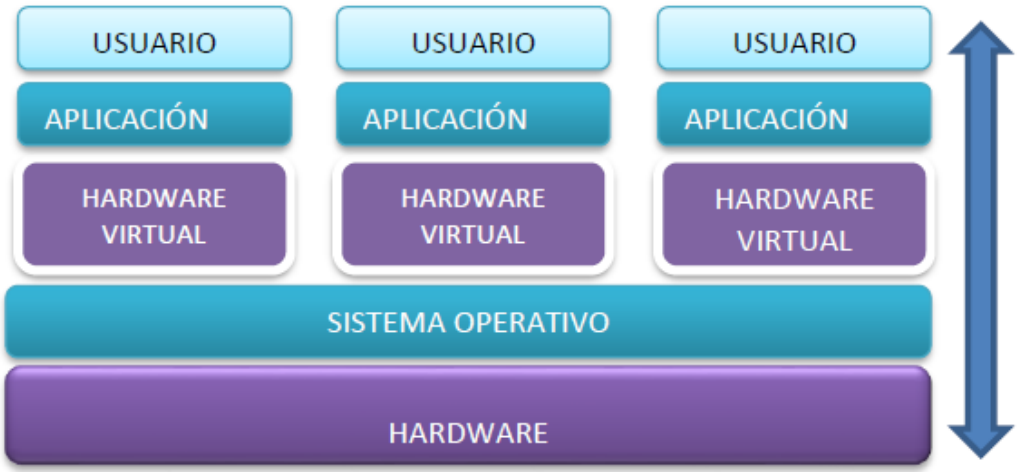
1.2. Objetivos.

Un sistema operativo es un conjunto de programas que, ordenadamente relacionados entre sí, contribuyen a que el ordenador lleve a efecto correctamente el trabajo encomendado.

Desde el punto de vista del usuario, el sistema operativo consiste en una serie de programas y funciones que ocultan los detalles del hardware, ofreciéndole una vía sencilla y flexible de acceso al mismo, teniendo dos objetivos fundamentales:

- **Seguridad:** El sistema operativo debe actuar contra cualquier manipulación extraña, ya sea accidental o premeditada que pudiera dañar la información, perjudicar a otros usuarios o provocar un funcionamiento indeseado del sistema. Por ejemplo, hay ciertas instrucciones que pueden parar la máquina y otras que realizan operaciones directamente sobre el hardware, que debemos evitar que se utilicen por los programas. Para ello, algunos sistemas proporcionan dos estados, llamados estado protegido (Sistema o **Kernel**), en el cual se ejecuta el **sistema operativo**, y estado no protegido (Usuario o **User**), que es el destinado a la ejecución de los **programas de usuario** y de aplicación. De esta manera se impide que los programas de los usuarios puedan tener contacto directo con el hardware, o puedan forzar un incorrecto funcionamiento del sistema.
- **Abstracción:** La tendencia actual del software y de los lenguajes de programación es ocultar lo más posible los detalles de más bajo nivel, intentando dar a los niveles superiores una visión más sencilla, global y abstracta, ofreciéndoles operaciones para manipular dichas estructuras ocultas, desconociendo por completo la gestión interna de

las mismas. Sobre estas estructuras se construyen otras que abstraen a las anteriores, y así sucesivamente. Gracias a la **abstracción**, los sistemas operativos **enmascaran** los **recursos físicos**, permitiendo su manejo con funciones más generales que ocultan las básicas, constituyendo verdaderos recursos ficticios o virtuales, que mejoran y son más potentes que los físicos.



Desde el punto de vista de un programa o usuario, la máquina física se convierte, gracias al sistema operativo, en una máquina virtual, también conocida como máquina extendida, que presenta la ventaja respecto a la física de ofrecer más funciones de las que normalmente soportaría esta última. Desde el punto de vista del usuario, el sistema operativo proporciona servicios que no están presentes en la máquina subyacente. Estos servicios incluyen las facilidades de carga y ejecución de programas, interacción entre el usuario y los programas, permitiendo que se ejecuten varios al mismo tiempo, gestión de la contabilidad para facturar los servicios y almacenamiento de datos y programas.

Como resumen, podemos decir que el sistema operativo persigue alcanzar la mayor eficiencia posible del hardware y facilitar el uso del mismo a los usuarios y a las aplicaciones.

2. Funciones de los sistemas operativos.

Las funciones de los sistemas operativos son diversas y han ido evolucionando de acuerdo con los progresos que la técnica y la informática han experimentado. Como principales funciones, podríamos enumerar las siguientes:

- Gestión de procesos. Hay que diferenciar entre los conceptos programa y proceso. Un programa es un ente pasivo, que cuando se carga en memoria y comienza a ejecutarse, origina uno o varios procesos.
- Gestión de la memoria. La gestión de memoria, suele ir asociada a la gestión de procesos. Para ejecutar un proceso es necesario asignarle unas direcciones de memoria exclusivas para él y cargarlo en ellas, cuando el proceso finalice su ejecución es necesario liberar las direcciones de memoria que estaba usando.
- Gestión de ficheros. Un fichero es una abstracción para definir una colección de información no volátil. Su objetivo es proporcionar un modelo de trabajo sencillo con la información almacenada en los dispositivos de almacenamiento. Estos ficheros deben tener espacio asignado en los dispositivos, deben estar protegidos entre ellos, deben organizarse según unos determinados esquemas… todo esto es la gestión de ficheros.
- Gestión de los dispositivos de E/S. La gestión de la entrada salida (E/S) tiene como objetivo proporcionar una interfaz de alto nivel de los dispositivos de E/S sencilla de utilizar.
- Gestión de la red. El sistema operativo es el encargado de gestionar los distintos niveles de red, los drivers (controladores) de los dispositivos involucrados en la red, los protocolos de comunicación, las aplicaciones de red, etc.
- Protección y seguridad. Mecanismos para permitir o denegar el acceso a los usuarios y a sus procesos a determinados recursos (ficheros, dispositivos de E/S, red, etc.).

Para comprender mejor porqué existen dichas funciones y cuáles son sus objetivos, las iremos estudiando mientras hacemos un breve recorrido a través de la historia de los ordenadores y la informática, ya que nos ayudara a comprender mejor el concepto de sistema operativo.

Los objetivos fundamentales de los sistemas operativos respecto a conseguir la mayor eficiencia y facilidad de uso posibles, no son siempre compatibles, ya que cualquier sistema que deba ser eficiente, normalmente no será fácil de usar, mientras que, si es fácil de usar, se deberá ofrecer a los usuarios muchas facilidades y ayudas, incluyendo muchos pasos e información que para un usuario experto no serían necesarias, lo que implica, obviamente, una pérdida de eficiencia.

3. Tipos de Sistemas operativos.

En este punto vamos a describir algunas características que permiten clasificar a los sistemas operativos. Básicamente veremos tres clasificaciones: sistemas operativos por su estructura (visión interna), sistemas operativos por los servicios que ofrecen y sistemas operativos por la forma en que ofrecen sus servicios (visión externa).

3.1. Sistemas operativos por su estructura (visión interna).

Si estudiamos los sistemas operativos atendiendo a su estructura interna, veremos que existen dos tipos fundamentales, los sistemas de estructura monolítica y los sistemas de estructura jerárquica.

En los sistemas operativos de estructura monolítica nos encontramos con que el sistema operativo está formado por un único programa dividido en rutinas, en donde cualquier parte del sistema operativo tiene los mismos privilegios que cualquier otra. Estos sistemas tienen la ventaja de ser muy rápidos en su ejecución (solo hay que ejecutar un programa) pero cuentan con el inconveniente de carecer de la flexibilidad suficiente para soportar diferentes ambientes de trabajo o tipos de aplicaciones. Es por esto que estos sistemas operativos suelen ser hechos a medida, para solucionar un problema en concreto y no para trabajar de forma generalista.

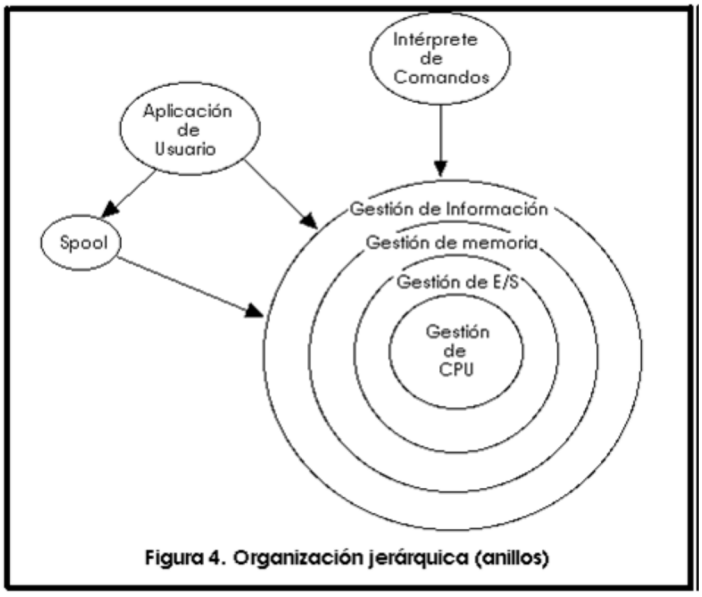
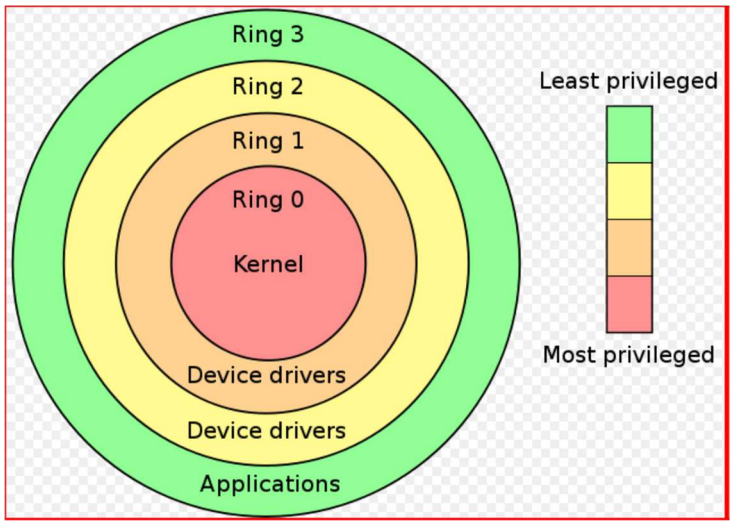


Figura 4. Organización jerárquica (anillos)

A medida que fueron creciendo las necesidades de los usuarios y se perfeccionaron los sistemas, se hizo necesaria una mayor organización del software, del sistema operativo, donde una parte del sistema contenía partes más pequeñas y esto organizado en forma de niveles. Se dividió el sistema operativo en pequeñas partes, de tal forma que cada una de ellas estuviera perfectamente definida y con un claro interface con el resto de elementos.

Se puede pensar también en estos sistemas como si fueran `multicapa'.



En la estructura anterior se basan prácticamente la mayoría de los sistemas operativos actuales. Otra forma de ver este tipo de sistema es la denominada de anillos concéntricos o "rings"

En el sistema de anillos, cada uno tiene una apertura, conocida como puerta o trampa (trap), por donde pueden entrar las llamadas de las capas inferiores. De esta forma, las zonas más internas del sistema

operativo o núcleo del sistema estarán más protegidas de accesos indeseados desde las capas más externas. Las capas más internas serán, por tanto, más privilegiadas que las externas.

Cada capa supervisa a la capa que tiene por encima, de modo que para que algo se ejecute en la capa 5, por ejemplo, debe recibir permiso y supervisión de la capa 4, que esta supervisada por la 3, y así sucesivamente. Evidentemente cuanto más al "exterior" de la estructura se ejecute un programa, más lento va a ser su funcionamiento ya que va a recibir un gran número de supervisiones. Por el contrario, cuanto más en el interior se ejecute un proceso, mayor será su velocidad.

En el centro de esta estructura se encuentra el Kernel o Núcleo del sistema operativo, que es su parte más importante.

3.2. Tipos de Sistemas Operativos según su visión interna.

Según el número de usuarios que soporta concurrentemente:

- Monousuarios. Los sistemas operativos monousuarios son aquéllos que soportan a un usuario a la vez, sin importar el número de procesadores que tenga la computadora o el número de procesos o tareas que el usuario pueda ejecutar en un mismo instante de tiempo. Las computadoras personales típicamente se han clasificado en esta sección.
- Multiusuario. Los sistemas operativos multiusuario son capaces de dar servicio a más de un usuario a la vez, ya sea por medio de varias terminales conectadas a la computadora o por medio de sesiones remotas en una red de comunicaciones. No importa el número de procesadores en la máquina ni el número de procesos que cada usuario puede ejecutar simultáneamente.

Según el número de tareas que puede ejecutar concurrentemente:

- Monotareas. Los sistemas monotarea son aquellos que sólo permiten una tarea a la vez por usuario. Puede darse el caso de un sistema multiusuario y monotarea, en el cual se admiten varios usuarios al mismo tiempo, pero cada uno de ellos puede estar haciendo solo una tarea a la vez.
- Multitareas. Un sistema operativo multitarea es aquél que le permite al usuario estar realizando varias labores al mismo tiempo. Por ejemplo, puede estar editando el código fuente de un programa durante su depuración mientras compila otro programa, a la vez que está recibiendo correo electrónico en un proceso en background (segundo plano). Es común encontrar en ellos interfaces gráficas orientadas al uso de menús y el ratón, lo cual permite un rápido intercambio entre las tareas para el usuario, mejorando su productividad.

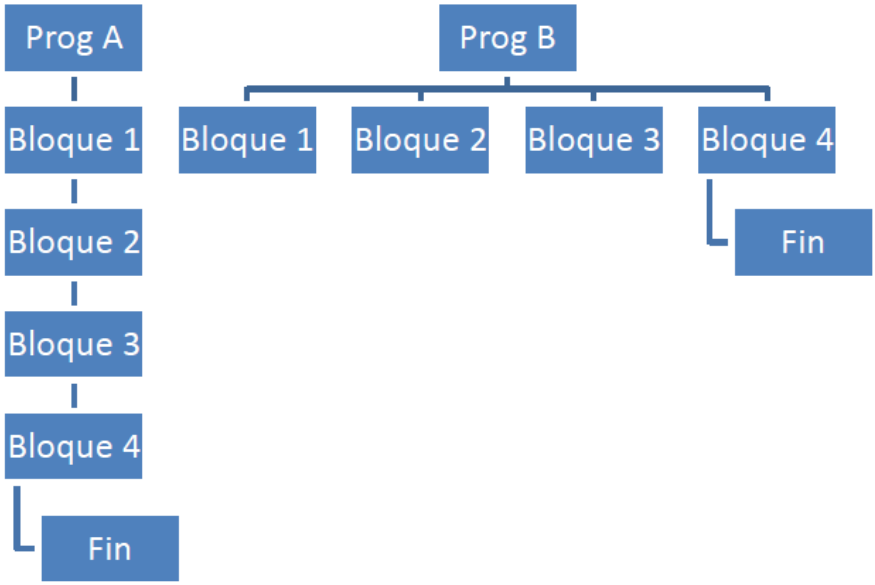
Según el número de procesadores que puede gestionar:

- Uniproceso. Un sistema operativo uniproceso es aquél que es capaz de manejar solamente un procesador de la computadora, de manera que si la computadora tuviese más de uno le sería inútil. Por ejemplo, Windows 98 es un sistema operativo Uniproceso.
- Multiproceso. Un sistema operativo multiproceso es capaz de manejar más de un procesador en el sistema, distribuyendo la carga de trabajo entre todos los procesadores que existan en el sistema. Generalmente estos sistemas trabajan de dos formas: simétricamente o asimétricamente.
    - Cuando se trabaja de manera asimétrica, el sistema operativo selecciona a uno de los procesadores el cual jugará el papel de procesador maestro y servirá como pivote para distribuir la carga a los demás procesadores, que

reciben el nombre de esclavos. El único procesador que realmente tiene acceso a todos los recursos del sistema es el maestro, que relega en los esclavos los trabajos que le van llegando. Es un sistema simple de construir y donde es muy fácil añadir más procesadores esclavos.

- Cuando se trabaja de manera simétrica, los procesos o partes de ellos (**threads, hebras o hilos**) son enviados indistintamente a cualquiera de los procesadores disponibles, teniendo una mejor distribución y equilibrio en la carga de trabajo bajo este esquema. Se dice que un thread es la parte activa en memoria y corriendo de un proceso, lo cual puede consistir de un área de memoria, un conjunto de registros con valores específicos, la pila y otros valores de contexto. Es un sistema mucho más difícil de construir, y es también tremendamente complicado añadir más procesadores, pero tiene la gran ventaja de ser muchísimo más práctico, ya que cada procesador tiene acceso a todos los recursos y las cargas de trabajo se pueden dividir de forma mucho más rápida.

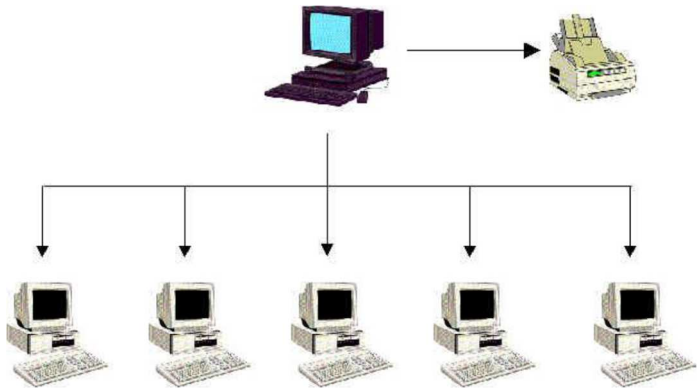
Un aspecto importante a considerar en estos sistemas es que la aplicación debe construirse específicamente para aprovechar varios procesadores. Existen aplicaciones que fueron hechas para correr en sistemas uniproceso que no aprovechan el multiproceso, ya que el código debe contener secciones de código paralelizable (que se puedan correr en paralelo), los cuales son ejecutados al mismo tiempo en procesadores diferentes.



Así, Prog A se ejecutará en 4 ciclos de reloj sin importar si se ejecuta en un sistema monoprocesador o no, mientras que Prog B se ejecutará en 4 ciclos de reloj en un sistema monoprocesador, pero en 1 ciclo de reloj en un sistema con 4 procesadores.

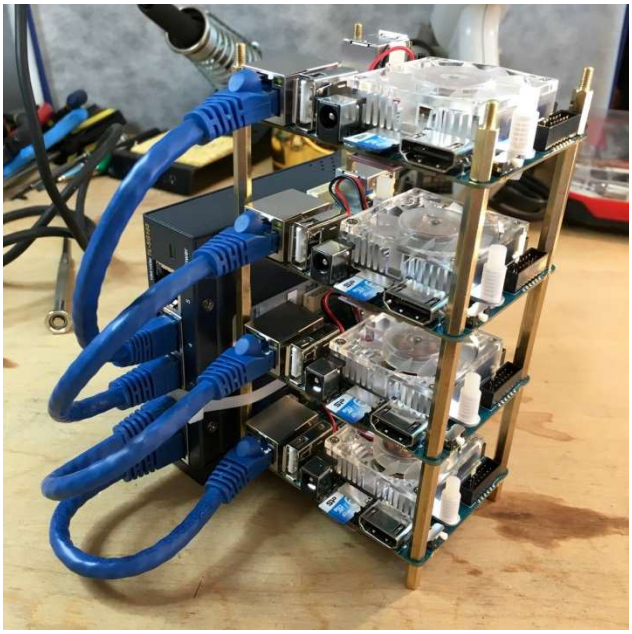
3.3. Sistemas Operativos según su visión externa.

Esta clasificación se refiere a la visión externa del usuario, en cómo este usuario accede a los servicios.



- SISTEMAS OPERATIVOS DE ESCRITORIO. Estos sistemas operativos se utilizan en los equipos personales, estaciones de trabajo, portátiles, etc. También se suelen conocer como sistemas operativos clientes. Windows 7 por ejemplo, es un sistema operativo de escritorio. Suelen ser sistemas operativos preparados para permitir un uso fácil por parte del usuario, destacan en multimedia, juegos, sonido, ofimática, etc.
- SISTEMAS OPERATIVOS EN RED Y SERVIDORES. Los sistemas operativos de red se definen como aquellos que tiene la capacidad de interactuar con sistemas operativos en otras computadoras por medio de un medio de transmisión con el objeto de intercambiar información, transferir archivos, ejecutar comandos remotos y un sin fin de otras actividades. Lo importante es hacer ver que el usuario puede acceder a la información no solo de su máquina, sino a la de cualquier máquina de la red, y esto se consigue gracias a que utiliza un sistema operativo de red.
  Hoy en día todos los sistemas operativos de escritorio son sistemas operativos de red también, cosa que no ocurría anteriormente. Normalmente solemos llamar sistemas operativos en red a los sistemas operativos que funcionan como servidores en una red, como es el caso del Windows Server o Linux Server.
- SISTEMAS OPERATIVOS DISTRIBUIDOS. Un sistema distribuido se define como una colección de equipos informáticos separados físicamente y conectados entre sí por una red de comunicaciones distribuida; cada máquina posee sus componentes de hardware y software de modo que el usuario percibe que existe un solo sistema (no necesita saber qué cosas están en qué máquinas). El usuario accede a los recursos remotos de la misma manera en que accede a recursos locales ya que no percibe que existan varios ordenadores, sino que solo es capaz de ver uno formado por todos los anteriores.
  Una ventaja fundamental de los sistemas distribuidos, es que permiten aumentar la potencia del sistema informático, de modo que 100 ordenadores trabajando en conjunto, permiten formar un único ordenador que sería 100 veces más potente que un ordenador convencional.
  Los sistemas distribuidos son muy confiables, ya que si un componente del sistema se descompone otro componente debe de ser capaz de reemplazarlo, esto se denomina Tolerancia a Fallos.
  El tamaño de un sistema distribuido puede ser muy variado, ya sean decenas de hosts (red de área local), centenas de hosts (red de área metropolitana), y miles o millones de hosts (Internet); esto se denomina escalabilidad. De hecho, si un ordenador formando por un sistema distribuido se queda “corto” para las necesidades de la empresa, basta con instalar más.
  La computación distribuida ha sido diseñada para resolver problemas demasiado grandes para cualquier supercomputadora y mainframe, mientras se mantiene la flexibilidad de trabajar en múltiples problemas más pequeños. Esta forma de computación se conoce como grid.

Los grandes retos de cálculo de hoy en día, como el descubrimiento de medicamentos, simulación de terremotos, inundaciones y otras catástrofes naturales, modelización del clima/tiempo, grandes buscadores de internet, el programa SETI, etc. Son posibles gracias a estos sistemas operativos distribuidos que permiten utilizar la computación distribuida.



El modelo de computación de ciclos redundantes, también conocido como computación zombi, es el empleado por aplicaciones como Seti@Home, consistente en que un servidor o grupo de servidores distribuyen trabajo de procesamiento a un grupo de computadoras voluntarias a ceder capacidad de procesamiento no utilizada. Básicamente, cuando dejamos nuestro ordenador encendido, pero sin utilizarlo, la capacidad de procesamiento se desperdicia por lo general en algún protector de pantalla, este tipo de procesamiento distribuido utiliza nuestra computadora cuando nosotros no la necesitamos, aprovechando al máximo la capacidad de procesamiento. La consola PS3 también cuenta con una iniciativa de este tipo.

Otro método similar para crear sistemas de supercomputadoras es el clustering. Un cluster o racimo de computadoras consiste en un grupo de computadoras de relativo bajo costo conectadas entre sí mediante un sistema de red de alta velocidad (gigabit de fibra óptica por lo general) y un software que realiza la distribución de la carga de trabajo entre los equipos. Por lo general, este tipo de sistemas cuentan con un centro de almacenamiento de datos único. Los clusters tienen la ventaja de ser sistemas redundantes, si falla un equipo se resiente un poco la potencia del cluster, pero los demás equipos hacen que no se note el fallo.

En un cluster normalmente todos los equipos están ubicados en una misma red de área local, mientras que en un grid los equipos suelen estar distribuidos por todo el mundo.

Algunos sistemas operativos que permiten realizar clustering o grid, son; Amoeba, BProc, DragonFly BSD, Génesis, Kerrighed, Mosix/OpenMosix, Nomad, OpenSSI, Plurid.

Un cluster que usamos habitualmente, es el que forma Google. Se estima que en 2010 usaba unos 450.000 ordenadores, distribuidos en varias sedes por todo el mundo y formando clusters en cada una de dichas sedes.

Cada cluster de Google está formado por miles de ordenadores y en los momentos en que se detecta que el sistema está llegando al límite de su capacidad, se instalan cientos de ordenadores más en pocos minutos, aumentado así la potencia de cada cluster. Estos equipos normalmente con ordenadores x86 como los que solemos usar nosotros, corriendo versiones especiales de Linux, modificadas por la propia Google para que permitan la formación de estos clusters. (Buscar en google “google centro de datos” para ver algunas informaciones sobre ellos).

3.4. Sistemas Operativos por su disponibilidad.

Dividimos aquí los sistemas operativos por la forma en que se ponen disponibles a los usuarios.

- SISTEMAS OPERATIVOS PROPIETARIOS.
  Se les denomina propietarios porque son sistemas propiedad de la empresa que los desarrolla. La empresa no vende en realidad el sistema operativo, sino una licencia de uso del mismo. No se tiene acceso al código fuente del sistema, o por lo menos, no se tiene permiso para modificarlo libremente.

También está prohibido distribuir estos sistemas, o usarlos de formas no autorizadas por la empresa desarrolladora. Toda la familia Windows es un claro ejemplo de sistema operativo propietario.

- SISTEMAS OPERATIVOS LIBRES.
  Son sistemas operativos en los que se ha renunciado a cualquier tipo de propiedad intelectual. Son sistemas que pueden usarse libremente, ser distribuidos, permiten que se acceda a su código fuente y permiten que esté sea modificado de la forma que queramos.
  No hay que confundir el hecho de que sean libres con el hecho de que sean gratuitos. Posteriormente trataremos en profundidad el tema de las licencias de software.
  En general, tanto los sistemas operativos como las aplicaciones normales, pueden definirse según su disponibilidad en alguno de estos apartados:

3.5. Sistemas Operativos por su tipo de licencia.

Dentro de los sistemas operativos comerciales, propietarios y privativos, nos podemos encontrar con diversos tipos de licencia de uso:

- O.E.M.
  OEM (abreviatura del inglés original equipment manufacturer, en español sería fabricante de equipamiento original). Este tipo de licencias se las otorga el desarrollador del sistema operativo al fabricante de hardware, de modo que cuando nosotros compramos uno de sus productos, este viene con una licencia de uso del sistema operativo de tipo OEM. La particularidad de este tipo de licencias, es el que el sistema operativo viene preparado para ese hardware específicamente, de modo que no tenemos realmente una licencia de uso del sistema operativo, sino una licencia de uso del sistema operativo únicamente para ese hardware en concreto.
  Estas licencias son las más económicas, y suelen poseer restricciones especiales, aparte de venir sin manuales ni caja.
- RETAIL.
  Es la licencia que compramos directamente del desarrollador. Somos propietarios de la licencia, podemos instarlo en cualquier tipo de hardware compatible, podemos revender la licencia o cederla, etc.
  Normalmente solo permiten su uso en una sola maquina a la vez. Vienen con su caja y manuales.
  En las licencias de tipo retail, normalmente podemos elegir entre una licencia completa, o una licencia de actualización, que permite actualizar un sistema anterior al nuevo, por un coste algo más reducido.
- VLM (LICENCIAS POR VOLUMEN).
  Para una empresa con cientos de ordenadores, es complicado controlar las licencias individuales de cada una de sus máquinas. Existe la posibilidad de contratar un tipo de licencia especial con el desarrollador, de modo que con una única clave de licencia,

podemos utilizar varias máquinas a la vez. Es habitual que existan licencias de 25 usos concurrentes, 50, etc.
Son las licencias más caras evidentemente, aunque son bastante más económicas que comprar cada una de las licencias individualmente.

- MSDN (LICENCIAS DE EDUCACIÓN.)
  Son unas licencias especiales de Microsoft que permiten su uso únicamente para actividades educativas y de formación. Cualquier uso de estas licencias en equipos que desarrollen actividades fuera de este ámbito, es ilegal. Existen también licencias de este tipo para empresas de desarrollo, academias, etc.

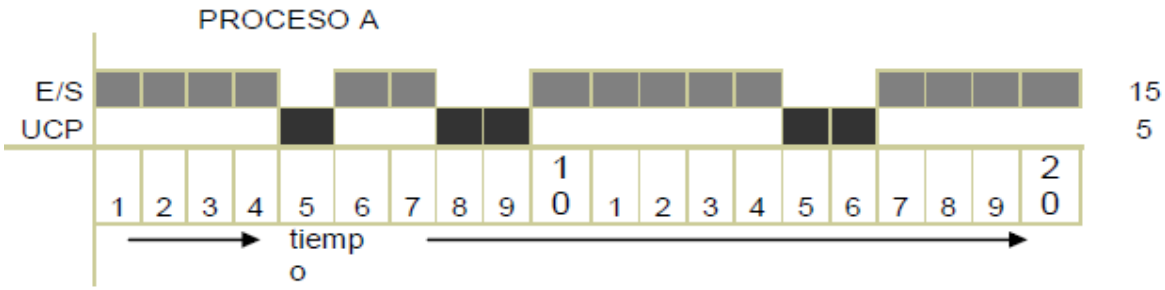
3.6. Según su tipo de respuesta o modo de explotación.

- Procesamiento por lotes (batch). Los trabajos se agrupan en bloques o lotes semejantes sin que exista interacción entre el usuario y los procesos mientras estos se ejecutan secuencialmente.
- Sistemas de tiempo compartido. El sistema se encarga de distribuir los procesos en función de un tiempo asignado de utilización del procesador o procesadores hasta la finalización del mismo.
- Sistemas en tiempo real. El tiempo de respuesta es inmediato para la solicitud de ejecución de un proceso. Se utilizan en entornos industriales como puede ser el CAN bus.

4. Administración de Procesos.

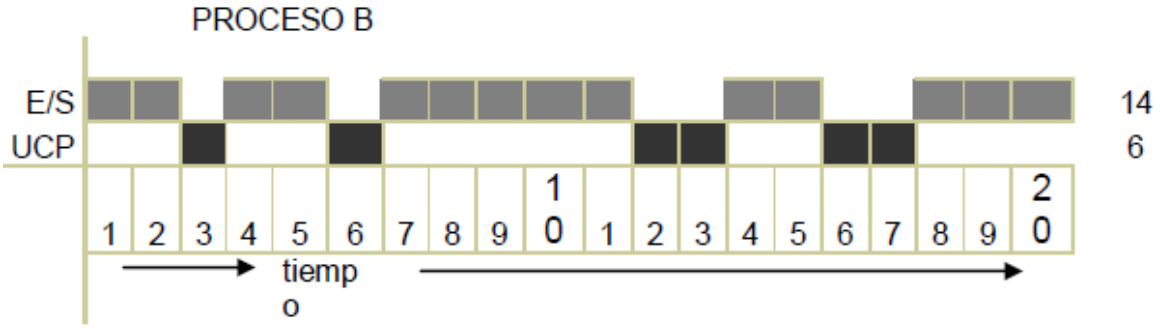
Si ejecutamos un solo programa en un ordenador, difícilmente podremos alcanzar un rendimiento del 100% ya que siempre tendrá que realizar operaciones de entrada/salida. Es decir, habrá tiempos muertos del procesador durante los que no realizará ningún trabajo, y no todo el tiempo estará realizando cálculos del programa. Esto es así porque las unidades de E/S (impresoras, monitores, teclados, etc.) son millones de veces más lentas que la CPU del ordenador.

Para comprenderlo mejor, podemos tomar como ejemplo la ejecución del programa representado en la figura siguiente, donde se detalla el diagrama de tiempos de ocupación de los recursos, incluido el propio procesador, necesarios para que puedan realizar el trabajo para el que fue diseñado.



En este primer proceso (A), vemos que se necesitan 20 unidades de tiempo para ejecutar en su totalidad el proceso, de las cuales 15 se van a usar para emplear los dispositivos de E/S (impresoras, discos, etc.) y 5 van a usarse para cálculos y procesos con la CPU. Establezcamos ahora un segundo proceso (B).

Vemos que este proceso (B) necesita también 20 unidades de tiempo para ejecutarse, de las cuales 14 van a emplearse para trabajar con las E/S, y 6 van a utilizarse para trabajar con la CPU

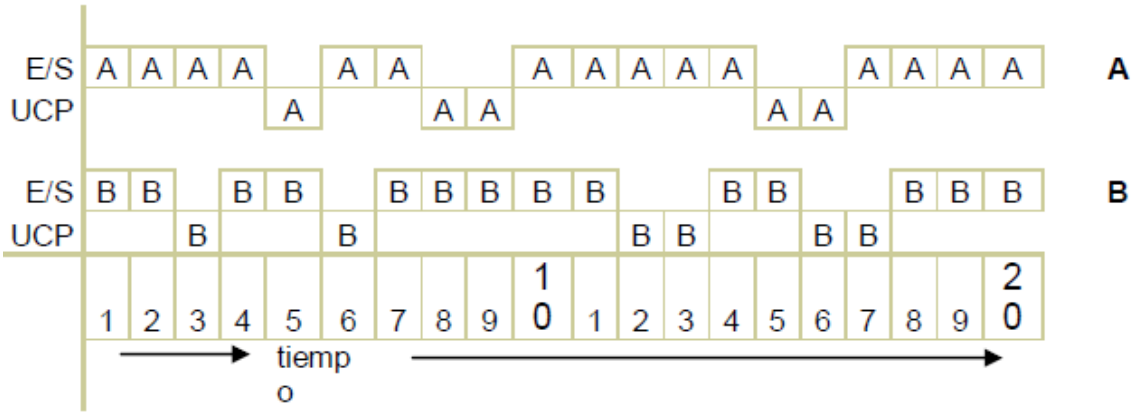


Si tenemos que ejecutar en nuestra máquina, el proceso A y luego el proceso B, el tiempo total de la ejecución será de 40 unidades de tiempo obviamente. Sin embargo… ¿no sería posible optimizar algo este tiempo?.

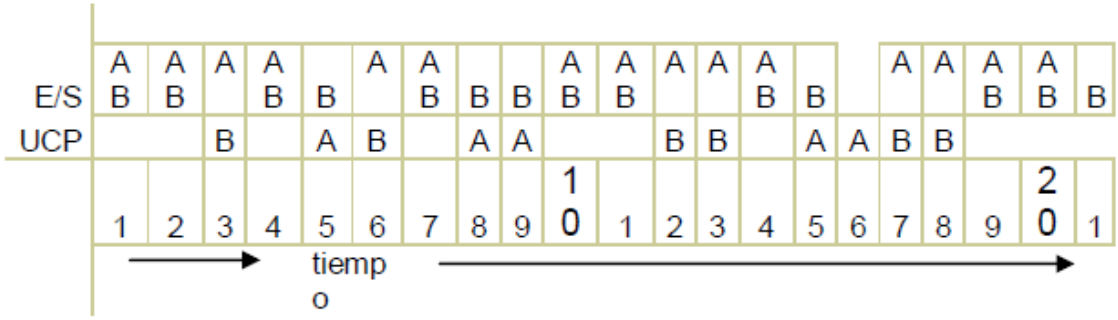
Si estudiamos el proceso A, veremos que durante las 4 primeras unidades de tiempo, la CPU está esperando, sin hacer absolutamente nada. Y podemos comprobar cómo el proceso B necesita usar la UCP en la unidad de tiempo 3.

Normalmente, las operaciones de E/S pueden ser concurrentes, es decir, desarrollarse al mismo tiempo, ya que pueden tratarse de operaciones con distintos dispositivos, o aprovechar las características de caché de los mismos.

Entonces, podemos ejecutar los dos procesos concurrentemente, alternando entre uno y otro, según la UCP vaya quedando libre, y solapando las operaciones de entrada salida. Vamos a ver como quedarían los anteriores trabajos si los ejecutamos concurrentemente, aprovechando los tiempos muertos. Recordemos las necesidades de ambos procesos.



Vemos aquí las necesidades de los procesos A y B, y podemos comprobar cómo muchas veces, B necesita la CPU y puede usarla porque A no la está usando, hagamos esta operación:



Aquí tenemos la ejecución de ambos procesos, realizando de forma concurrente las operaciones de entrada salida, y aprovechando los tiempos muertos de la CPU. Fijaros como el tiempo total para ejecutar los dos procesos es de 21 unidades de tiempo, solo 1 más que lo que lleva ejecutar un solo proceso, y muchísimo menos de las 40 u. que llevaría ejecutar ambos procesos de forma independiente.

Además, podemos comprobar cómo gracias a estas técnicas, aprovechamos al máximo los componentes de nuestro sistema, ya que las unidades de E/S están usándose en 20 de las 21 unidades de tiempo, y la CPU está usándose al menos 11 de las 21 unidades de tiempo.

Esta técnica se conoce como multiprogramación y tiene como finalidad conseguir un mejor aprovechamiento de los recursos del ordenador, ejecutando simultáneamente varios programas ofreciendo una falsa apariencia de ejecución paralela de los mismos. (Como hemos visto, la ejecución en la CPU es de un proceso al mismo tiempo, no pudiendo ejecutar los 2 procesos a la vez).

Si contamos con un microprocesador que tenga varias CPU, ahí sí sería posible una ejecución paralela real (de ahí la existencia de microprocesadores con varios núcleos, o lo que es lo mismo, varias CPU encapsuladas juntas).

La utilización de la multiprogramación ha dado lugar a diferenciar los trabajos de acuerdo con sus características y necesidades de recursos, pudiendo clasificarlos en dos tipos.

En primer lugar los trabajos limitados por proceso, es decir, aquellos que consumen la mayor parte de su tiempo en el tratamiento de la información y poco en entrada / salida.

En segundo lugar, los trabajos limitados por entrada/salida que basan su acción en operaciones de entrada/salida haciendo poco uso del procesador, el cual permanecerá la mayor parte del tiempo inactivo, considerándose como ideales desde el punto de vista de la multiprogramación.

Como hemos podido ver, la multiprogramación es una técnica altamente recomendable, pero al llevarla a cabo, nos vamos a encontrar con una serie de problemas que habrá que resolver. Vamos a ver algunos de estos problemas.

Uno de las funciones más importantes de un sistema operativo es la de administrar los procesos y tareas del sistema. Vamos a ver en este punto, dos temas relacionados con la administración de procesos; la planificación del procesador y los problemas de concurrencia.

4.1. Planificación del procesador.

Si recordamos cuando vimos la multiprogramación, vimos que en muchas ocasiones un proceso debía dejar paso a otro o esperarse cuando había conflictos en el uso de la CPU. Pues el planificador es el encargado de decidir qué proceso entra en cada momento.

La planificación del procesador se refiere a las técnicas que se usan para decidir cuánto tiempo de ejecución y cuando se le asignan a cada proceso del sistema en un sistema multiprogramado (multitarea). Obviamente, si el sistema es monoprogramado (monotarea) no hay mucho que decidir, pero en el resto de los sistemas esto es crucial para el buen funcionamiento del sistema.

Supóngase un ordenador que contiene un único microprocesador. Dicho microprocesador solamente puede ejecutar un programa en cada instante de tiempo. Además, cuando un programa está ejecutándose, nunca dejará de hacerlo por sí mismo. De manera que, en principio, cualquier programa monopoliza el microprocesador impidiendo que otros programas se ejecuten.

Por ello, la primera misión de un planificador es expulsar el programa en ejecución cuando decida que es pertinente. Esto se consigue de dos maneras, siempre con ayuda del propio hardware:

- Cuando expira un temporizador, que se activa a intervalos regulares de tiempo. En intervalos muy cortos, generalmente cada 250 milisegundos.
- Cuando el programa solicita una operación de entrada/salida. Dado que el programa no puede continuar hasta que termine dicha operación, es un buen momento para ejecutar otro programa.

En ambos casos, el control del microprocesador pasa a manos del planificador gracias a que el hardware genera una interrupción. En este proceso de expulsión, se guarda el estado de ejecución del programa (programa y su estado se denomina proceso).

A continuación, el planificador decide cuál será el siguiente proceso en ejecutarse. Naturalmente, solamente se escogen procesos que estén listos para hacerlo. Si un proceso sigue esperando por una operación de entrada/salida no será candidato a ejecutarse hasta que finalice tal operación.

La selección del proceso sigue alguna política de planificación preestablecida. Una vez seleccionado un proceso, se procede a ejecutarlo. Para ello, el planificador restaura su estado de ejecución (previamente salvado) y abandona el uso del microprocesador cediéndoselo a dicho proceso.

Todo este proceso se realiza en millonésimas de segundo.

Gracias a que el tiempo del microprocesador se reparte entre todos los procesos a intervalos muy cortos, el ordenador ofrece la sensación de que todos los procesos están ejecutándose a la vez.

Cuando un ordenador tiene varios microprocesadores este esquema se repite para cada microprocesador.

4.2. Problemas de Concurrencia.

En los sistemas de tiempo compartido (aquellos con varios usuarios, procesos, tareas, trabajos que reparten el uso de CPU entre estos) se presentan muchos problemas debido a que los procesos compiten por los recursos del sistema.

Imaginemos que un proceso está escribiendo en el disco duro y se le termina su turno de ejecución e inmediatamente después el proceso elegido para ejecutarse comienza a escribir sobre el disco duro. El resultado es un disco duro cuyo contenido es un desastre de datos mezclados. Así como el disco duro, existen una multitud de recursos cuyo acceso debe ser controlado para evitar los problemas de la concurrencia.

El sistema operativo debe ofrecer mecanismos para sincronizar la ejecución de procesos: semáforos, envío de mensajes, 'pipes', etc.

Los semáforos son rutinas de software (que en su nivel más interno se auxilian del hardware) para lograr exclusión mutua en el uso de recursos. Para entender este y otros mecanismos es importante entender los problemas generales de concurrencia, los cuales se describen enseguida.

- Condiciones de Carrera o Competencia: La condición de carrera (race condition) ocurre cuando dos o más procesos acceden a un recurso compartido sin control, de manera que el resultado combinado de este acceso depende del orden de llegada. Supongamos, por ejemplo, que dos personas realizan una operación en un banco en la

misma cuenta corriente, uno A por el cajero, y uno B mediante Internet desde su casa. El usuario A quiere hacer un depósito. El B un retiro. El usuario A comienza la transacción y lee su saldo que es 1000. En ese momento pierde su turno de ejecución por parte del ordenador del banco (y su saldo queda como 1000) y el usuario B inicia el retiro: lee el saldo que es 1000, retira 200 y almacena el nuevo saldo que es 800 y termina. El turno de ejecución regresa al usuario A el cual hace su depósito de 100, quedando saldo = saldo + 100 = 1000 + 100 = 1100. Como se ve, el retiro se perdió y eso será magnifico para los usuarios A y B, pero al banquero no le haría demasiada gracia. El error pudo ser al revés, quedando el saldo final en 800.

- Postergación o Aplazamiento Indefinido(a): Consiste en el hecho de que uno o varios procesos nunca reciban el suficiente tiempo de ejecución para terminar su tarea. Por ejemplo, que un proceso ocupe un recurso y lo marque como 'ocupado' y que termine sin marcarlo como 'desocupado'. Si algún otro proceso pide ese recurso, lo verá 'ocupado' y esperará indefinidamente a que se 'desocupe'.
- Condición de Espera Circular: Esto ocurre cuando dos o más procesos forman una cadena de espera que los involucra a todos. Por ejemplo, suponga que el proceso A tiene asignado el recurso 'pantalla' y el proceso B tiene asignado el recurso 'disco'. En ese momento al proceso A se le ocurre pedir el recurso 'disco' y al proceso B el recurso 'pantalla'. Ahí se forma una espera circular infinita entre esos dos procesos.

4.3. Procesos, flujos y hebras.

A los procesos, dependiendo del sistema operativo utilizado, se les denomina flujos de control, tareas, threads o hilos, dependiendo del contexto.

Cuando se ejecuta más de un proceso de forma concurrente en un sistema, todos necesitan que el propio sistema les suministre una serie de recursos. Para ello el sistema operativo se encarga de asignar estos recursos en un orden adecuado y atendiendo a unas prioridades.

Una hebra o hilo es un subproceso de un proceso que consume recursos propios pero que depende del proceso padre que lo ha ejecutado.

Las hebras representan un método software para mejorar el rendimiento y eficacia de los sistemas operativos. Las hebras de un mismo proceso compartirán recursos, como memoria, archivos, recursos hardware, etc.

4.4. Bloque de control de procesos.

Los sistemas operativos disponen de los servicios necesarios para la gestión de los procesos, tales como su creación, terminación, ejecución periódica, cambio de prioridad, etc.

Toda la información de un procesos que el sistema operativo necesita para controlarlo se mantiene en una estructura de datos denominada bloque de control de procesos (BCP).

Cada vez que un programa se convierte en proceso, es decir, cada vez que se ejecuta un programa, además de ubicar en memoria las instrucciones que lo componen y sus datos asociados, a dicho procesos se le asigna una BCP.

El BCP de cada proceso almacena información como:

1.1. Estado actual: si el proceso está en ejecución, preparado o bloqueado.
1.2. Identificador del proceso: A cada proceso se le asigna un número único que sirve para identificar el proceso. A este número se le llama PID.
1.3. Prioridad del proceso: dependiendo de la importancia del proceso el planificador le asigna una prioridad.
1.4. Ubicación en memoria: Dirección de memoria en la que se carga el proceso.
1.5. Recursos utilizados: Que recursos tanto hardware como software tiene asignados para poder ejecutarse.

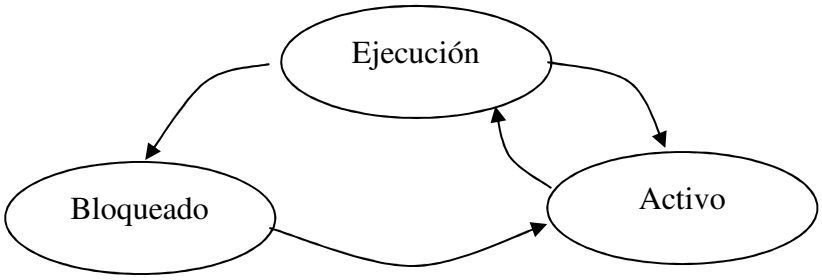
4.5. Estados de los procesos.

Básicamente los estados posibles de un proceso son 3 ejecución, preparado y bloqueado.

Ejecución: El procesador está ejecutando instrucciones del programa que lo compone y tiene concedido el tiempo de uso de la CPU en ese instante concreto.

Preparado: en espera o activo: El proceso está preparado para ser ejecutado, pero no dispone de tiempo de CPU para poder ejecutarse.

Bloqueado: El proceso está retenido por cualquier causa (normalmente espera un E/S), y hasta que esa no se termine, no puede continuar con la ejecución.



Entre estos 3 estados existen 4 transiciones que son:

- De ejecución a bloqueado: se produce cuando un proceso no puede seguir ejecutándose porque necesita de un dato, señal, para continuar ejecutándose. Normalmente espera en una operación de entrada salida.
- De ejecución a Activo: Se produce cuando al proceso, que tiene todo para poder ejecutarse se le quita la CPU, para asignársela a otro proceso.
- De bloqueado a activo: La operación que tenía bloqueado el proceso se ha completado, por lo tanto puede volver a ejecutarse.
- De activo a ejecución: el proceso tiene todo lo necesario para ejecutarse salvo la CPU, estará esperando para pasar a ejecución.

Cada vez que se produce un cambio, se produce un cambio de contexto. Los cambios de contexto tienen que ser operaciones livianas, ya que también consumen tiempo y afectaran al rendimiento del sistema.

El encargado de asignar recursos del sistema, y decidir qué proceso pasa a ejecutarse se llama planificador. El planificador tendrá en cuenta la importancia del proceso para asignarle una prioridad y por lo tanto priorizar en la asignación de recursos.

4.6. Algoritmos de planificación.

Existen 2 objetivos para planificar que proceso se ejecuta en un ordenador. El primero objetivo de la planificación es que se pueda simular que se ejecutan más de un proceso y el segundo realizar un uso eficiente del recurso que supone el tiempo de CPU, para evitar que el procesador este ocioso sin realizar ninguna tarea.

Un algoritmo de planificación deberá de determinar:
Cuando un proceso en ejecución pasa a activo.
Cuando un proceso en ejecución pasa a bloqueado.
Cuando un proceso en activo pasa a ejecución.
Cuando termina un proceso.

En caso 1 y 3 el algoritmo es expropiativo ya quita al proceso actual para colocar otro. En los casos 2 y 4 el algoritmo es no expropiativo, ya que hay que coger otro proceso para ser ejecutado.

Existen varios criterios que se pueden utilizar para determinar que algoritmo usar, estos pueden ser:

- % de utilización de CPU. Intentar tener ocupada la CPU el mayor tiempo posible.
- Rendimiento. Número de trabajos completados por unidad de tiempo.
- Tiempo de retorno. Tiempo transcurrido entre la llegada del proceso y su finalización.
- Tiempo de respuesta. Tiempo que tarda un proceso en ejecutarse desde que se produce un evento.

Los posibles objetivos de la planificación dependerán del ámbito, así se tiene:

- Minimizar el tiempo medio de espera.
- Maximizar la utilización de la CPU.
- Mantener el tiempo de respuesta por debajo de un valor.

No existe un algoritmo de planificación óptimo, porque cambiara dependiendo del

escenario como recursos, número de procesos, el orden de llegada, la prioridad de los procesos, etc.

En la planificación de procesos de basan en entornos deterministas, hecho que no es real, ya que suelen ser procesos interactivos (el usuario interactúa con el equipo), y no se sabe la duración del proceso, pero para simplificar su estudio se realiza determinista.

- Algoritmo FCFS (First Come First Serve).

En esta política de planificación, el procesador ejecuta cada proceso hasta que termina, por tanto, los procesos que en cola de procesos preparados permanecerán encolados en el orden en que lleguen hasta que les toque su ejecución. Este método se conoce también como FIFO (first input, first output, Primero en llegar primero en salir).

Se trata de una política muy simple y sencilla de llevar a la práctica, pero muy pobre en cuanto a su comportamiento.

La cantidad de tiempo de espera de cada proceso depende del número de procesos que se encuentren en la cola en el momento de su petición de ejecución y del tiempo que cada uno de ellos tenga en uso al procesador, y es independiente de las necesidades del propio proceso.

Sus características son:

– No expropiativo.
– Es justa, aunque los procesos largos hacen esperar mucho a los cortos.
– Predecible.
– El tiempo medio de servicio es muy variable en función del número de procesos y su duración.

- Algoritmo Round Robin o rueda.

Es un método para seleccionar todos los elementos en un grupo de manera equitativa y en un orden racional, normalmente comenzando por el primer elemento de la lista hasta llegar al último y empezando de nuevo desde el primer elemento.

Round Robin es uno de los algoritmos de planificación de procesos más complejos y difíciles, dentro de un sistema operativo asigna a cada proceso una porción de tiempo equitativa y ordenada, tratando a todos los procesos con la misma prioridad.

Se define un intervalo de tiempo denominado cuanto o quantum, cuya duración varía según el sistema. La cola de procesos se estructura como una cola circular. El planificado la recorre asignando un cuanto de tiempo a cada proceso. La organización de la cola es FIFO.

Este algoritmo es expropiativo.

- Algoritmo SJF (Shortest Job First).

En este algoritmo, da bastante prioridad a los procesos más cortos a la hora de ejecución y los coloca en la cola.

En resumen, este algoritmo selecciona al proceso con el próximo tiempo ejecución más corto. El proceso corto saltará a la cabeza de la cola. El algoritmo selecciona aquel proceso cuyo próximo ciclo de ejecución de CPU sea menor. El problema está en conocer dichos valores, pero podemos predecirlos usando la información de los ciclos anteriores ejecutados.

Este algoritmo no es expropiativo.

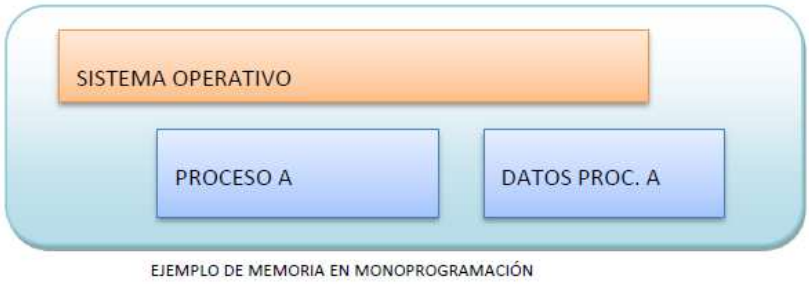
- Algoritmo STR (Short Time Remainder).

Es similar al SJF, con la diferencia de que si un nuevo proceso pasa a listo se activa el dispatcher (planificador) para ver si es más cortó que lo que queda por ejecutar del proceso en ejecución. Si es así, el proceso en ejecución pasa a listo y su tiempo de estimación se decremento con el tiempo que ha estado ejecutándose.

Este algoritmo es expropiativo.

5. Gestión de memoria.

En un sistema monoprogramado (lo contrario que multiprogramado), en la memoria del ordenador solo hay un único programa, acompañado de sus datos y del sistema operativo. Esto hace que el uso de la memoria, y la asignación de la misma al programa sea muy simple. Sin embargo, en un sistema multiprogramado nos vamos a encontrar en memoria con varios programas a la vez (2 en el mejor de los casos, pero podemos realizar multiprogramación con 20, 100 o 700 procesos).



EJEMPLO DE MEMORIA EN MONOPROGRAMACIÓN

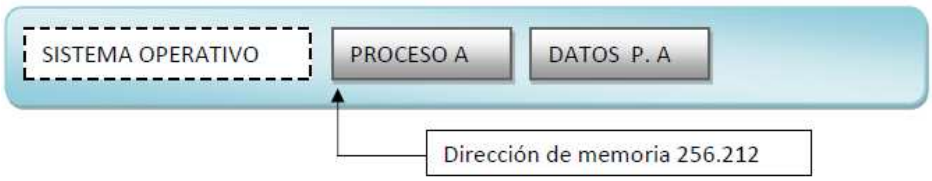
EJEMPLO DE MEMORIA EN MULTIPROGRAMACIÓN

Este "lio" que vemos en la memoria usando multiprogramación hace que se presenten dos problemas fundamentales, la relocalización y la protección.

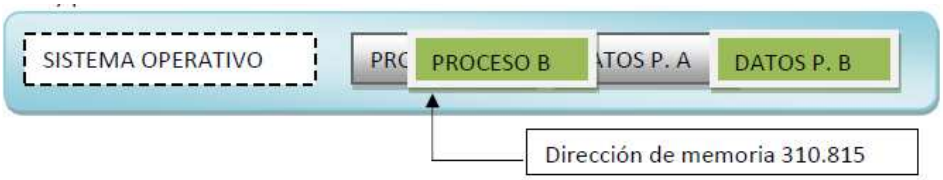
5.1. Problemas con la memoria. Relocalización.

Este problema consiste en que los programas que necesitan cargarse a memoria real ya están compilados y montados, de manera que internamente contienen una serie de referencias a direcciones de instrucciones, rutinas y procedimientos que ya no son válidas en el espacio de direcciones de memoria real de la sección en la que se carga el programa. Esto es, cuando se compiló el programa se definieron o resolvieron las direcciones de memoria de acuerdo a la sección de ese momento, pero si el programa se carga en otro día en una sección diferente, las direcciones reales ya no coinciden.

Si en memoria solo va a estar este programa, no hay problemas en cargarlo siempre en la misma dirección o sección de memoria, pero si cargamos varios programas en la memoria, esto ya no es posible, dado que varios programas podrían pedir la misma sección.



En este ejemplo teórico que vemos arriba, el proceso A está preparado para cargarse en la dirección de memoria 256.212. Mientras que solo dicho proceso esté funcionando en memoria no hay problema en esto.



En este ejemplo teórico de arriba, vemos como al mismo tiempo que el proceso A, intentamos ejecutar un proceso B que quiere cargarse en la dirección de memoria 310.815… vemos como "machaca" al proceso A, y también sus datos. Esto conllevaría que ambos procesos dejarían de funcionar, ya que irían sobrescribiéndose el uno al otro.

El gestor o controlador de memoria del sistema operativo, puede solucionar este problema de varias formas, una de las más usadas consiste en tener un registro que guarde la dirección base de la sección que va a contener al programa. Cada vez que el programa haga una referencia a una dirección de memoria, se le suma el registro base para encontrar la dirección real. Así, en el ejemplo anterior tendríamos que cuando el proceso A quiere acceder a la memoria para instalarse en la dirección 256.212, el SO (Sistema Operativo) hace la operación (dirección + base) donde base por ejemplo es 20.000 por lo que el proceso A sería cargado en la dirección 276.212. Cuando el proceso B se quiera cargar, el SO ajusta la base por ejemplo a 500.000 (para saltarse al proceso A), de forma que en lugar de instalarse en la dirección 310.815 se instalará en la dirección 810.815.

5.2. Problemas con la memoria. Protección.

Este problema se refiere a que, una vez que un programa ha sido cargado a memoria en algún segmento en particular, nada le impide al programador que intente direccionar (por error o deliberadamente) localidades de memoria menores que el límite inferior de su programa o superiores a la dirección mayor; es decir, quiere referenciar localidades fuera de su espacio de direcciones.

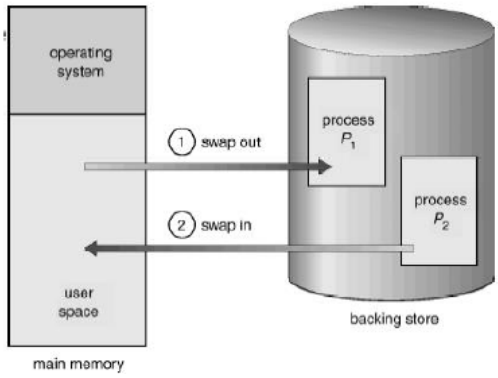
Obviamente, este es un problema de protección, ya que no es legal leer o escribir en áreas de memoria que pertenezcan a otros programas, y hay que proteger estas zonas de memoria.

La solución a este problema puede ser el uso de un registro base y un registro límite. El registro base contiene la dirección del comienzo de la sección que contiene el programa, mientras que el límite contiene la dirección donde termina. Cada vez que el programa hace una referencia a memoria se comprueba si cae en el rango de los registros y si no es así se envía un mensaje de error y se aborta el programa.

Muchos programas antiguos dan errores de acceso a memoria cuando son ejecutados en sistemas multiprogramados. Precisamente estos errores vienen dados por que intentan acceder a direcciones de memoria que quedan fuera de su ámbito.

En estos casos, el sistema operativo impide que dichos programas accedan a esas direcciones, lo que provoca un error en el acceso a memoria.

5.3. Memoria virtual.



Las CPU de los ordenadores eran cada vez más poderosas, lo que permitía ejecutar programas cada vez más potentes, y por lo tanto más grandes.

Al mismo tiempo, el uso de la multiprogramación, hacía que el sistema operativo tuviera que colocar decenas de estos grandes procesos en la memoria RAM.

El problema es que la cantidad de memoria RAM "física" que podemos instalar en un ordenador es finita, es decir, tiene un límite. Así, si intentamos ejecutar en multiprogramación 20 procesos, y cada uno de ellos necesita 512 MB de RAM para trabajar, tendríamos que tener instalados en nuestro ordenador 10 GB de RAM.

Todo esto empujó a los diseñadores de los sistemas operativos a implantar un mecanismo que permitiera ofrecer a los procesos más cantidad de memoria de la que realmente estaba instalada en la máquina, esto es, de ofrecer `memoria virtual'.

La memoria virtual se llama así porque el programa ve una cantidad de memoria mucho mayor que la real, y que en realidad se trata de la suma de la memoria de almacenamiento primario y una cantidad determinada de almacenamiento secundario (por ejemplo, el del disco duro).

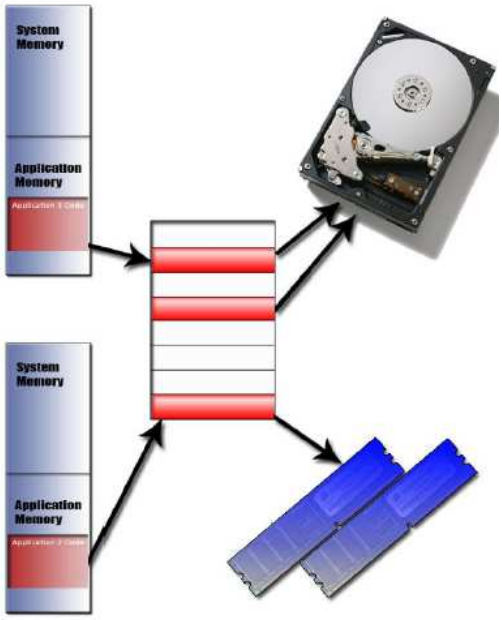
El sistema operativo, en su módulo de gestión de memoria, se encarga de intercambiar programas enteros, segmentos o páginas entre la memoria real y el medio de almacenamiento secundario. Si lo que se intercambia son procesos enteros, se habla entonces de multiprogramación en memoria real, pero si lo que se intercambian son segmentos o páginas, se puede hablar de multiprogramación con memoria virtual.

Existe una técnica en la cual, el sistema operativo divide a los procesos en pequeñas porciones, de tamaño fijo denominadas páginas, de un tamaño múltiplo de 1 K. Estas páginas van a ser pasadas de la RAM al disco y viceversa. Al proceso de intercambiar páginas, segmentos o programas completos entre RAM y disco se le conoce como `intercambio' o `swapping'.

En la paginación, se debe cuidar el tamaño de las páginas, ya que si éstas son muy pequeñas el control por parte del sistema operativo para saber cuáles están en RAM y cuales, en disco, sus direcciones reales, etc. crece y provoca mucha `sobrecarga' (overhead). Por otro lado, si las páginas son muy grandes, el overhead disminuye, pero entonces puede ocurrir que se desperdicie memoria en procesos pequeños. Debe haber un equilibrio entre ambos conceptos.

Otro aspecto importante es la estrategia para cargar páginas (o segmentos) a la memoria RAM. Se usan más comúnmente dos estrategias: cargado de páginas por demanda y cargado de páginas anticipada. La estrategia de cargado por demanda consiste en que las páginas solamente son llevadas a RAM si son solicitadas, es decir, si se hizo referencia a una dirección que cae dentro de ellas.

La carga anticipada consiste en tratar de adivinar qué páginas serán solicitadas en el futuro inmediato y cargarlas de antemano, para que cuando se pidan ya no ocurran fallos de página. Ese `adivinar' puede ser que se aproveche del fenómeno de localidad y que las páginas que se cargan por anticipado sean aquellas que contienen direcciones contiguas a la dirección que se acaba de referenciar. En el caso de Windows, se usa una conjunción de ambos métodos, y se utiliza un fichero en disco duro donde se va almacenando toda la información sobre índices de páginas en HD, en RAM, etc. Este fichero se denomina pagefile.sys y lo podéis encontrar normalmente en la raíz de vuestro volumen de sistema.



En los sistemas operativos Linux, la memoria virtual se implementaba mediante una partición destinada exclusivamente a ello, llamada partición swap o área de intercambio. Actualmente utiliza dispositivos virtuales dentro del sistema donde está ubicado el sistema.

Un problema que tenemos con la memoria virtual, es la diferencia de velocidad enorme que existe entre la RAM y la memoria de almacenamiento secundario. Si cargamos muchos procesos, y agotamos nuestra memoria RAM real, el SO permitirá que todo siga funcionando usando el HD, pero tardará muchísimo en pasar las páginas de RAM a HD y viceversa. En muchas ocasiones, nos parecerá incluso que el SO se ha quedado “colgado”. Esto es muy habitual en sistemas con poca memoria, donde vemos que de repente la luz indicadora de actividad en los discos duros se queda encendida, y el SO deja de responder durante un buen rato.

Un intento de solucionar esto es usar memorias de estado sólido en lugar del HD para paginar ya que son memorias mucho más rápidas, esto fue implementado desde Windows Vista.

6. Gestión de Datos. Sistemas de Ficheros.

Un fichero es un mecanismo de abstracción que sirve como unidad lógica de almacenamiento de información. El fichero agrupa una colección de informaciones relacionadas entre sí y definidas por su creador. A todo fichero le corresponde un nombre único que lo identifique entre los demás ficheros.

Es necesario que el sistema operativo cuente con un sistema que se encargue de administrar la información almacenada en los dispositivos en forma de ficheros: de esto se encargan los sistemas de ficheros.

Un sistema de ficheros es el aspecto más visible de todo sistema operativo y existe por razones tecnológicas, ya que no hay memoria principal lo suficientemente grande como para no necesitar de almacenamiento secundario. Surge debido a la necesidad del sistema operativo de poder gestionar la información de forma eficiente y estructurada, además de establecer unos parámetros de seguridad y protección en entornos críticos.

El sistema operativo ofrece una visión lógica y uniforme del almacenamiento de información realizando una abstracción de las propiedades físicas de sus dispositivos de almacenamiento. Para ello, define el concepto lógico de fichero. El sistema operativo se debe encargar del acoplamiento entre los ficheros y los dispositivos físicos, por medio del sistema de ficheros, que debe ser independiente del soporte físico concreto sobre el que se encuentre.

Los objetivos principales de todo sistema de ficheros deben ser los siguientes:

- Crear, borrar y modificar ficheros.
- Permitir el acceso controlado a la información.
- Permitir intercambiar datos entre ficheros.
- Poder efectuar copias de seguridad recuperables.
- Permitir el acceso a los ficheros mediante nombres simbólicos.

Hay otros objetivos secundarios, entre los que destacan:

- Optimizar el rendimiento.
- Tener soportes diversos para E/S (para poder seguir utilizando los mismos ficheros, aunque cambie el soporte).
- Ofrecer soporte multiusuario.
- Minimizar las pérdidas de información.

Normalmente los ficheros se organizan en directorios (también llamados carpetas) para facilitar su uso. Estos directorios son ficheros que contienen información sobre otros ficheros: no son más que contenedores de secuencias de registros, cada uno de los cuales posee información acerca de otros ficheros.

La información que contiene un fichero la define su creador. Hay muchos tipos diferentes de información que puede almacenarse en un fichero: programas fuente, programas objeto, datos numéricos, textos, registros contables, fotografías, videos, etc. Un fichero tiene una cierta estructura, definida según el uso que se vaya a hacer de él. Por ejemplo, un fichero de texto es una secuencia de caracteres organizados en líneas (y posiblemente en páginas); un fichero fuente es una secuencia de subrutinas y funciones, un fichero gráfico es una secuencia que permite dibujar pixeles en pantalla, etc.

Cualquier sistema operativo distingue entre varios tipos básicos de ficheros, que será la clasificación que consideremos nosotros:

- Regulares o Normales: Aquellos ficheros que contienen datos (información).
- Directorios: Aquellos ficheros cuyo contenido es información sobre otros ficheros, normalmente un vector de entradas con información sobre los otros ficheros.
- De dispositivo: Existen dispositivos cuya E/S se realiza como si fuesen ficheros, por lo tanto, es razonable asociarles ficheros para simplificar y hacer más transparente el intercambio de información con dichos dispositivos.

Aunque se imponga al sistema operativo el desconocimiento del tipo de ficheros que manipula, sí se hace una distinción del mismo de forma transparente: a través de las extensiones del nombre. Mediante la extensión del nombre del fichero (una cadena de caracteres de pequeña longitud) se puede determinar el tipo del fichero. Algunos sistemas de ficheros consideran a la extensión como una parte del nombre (y, de hecho, admiten que un mismo fichero posea varias extensiones anidadas), y otros la diferencian del nombre a nivel interno.

De este modo, aunque el sistema operativo no conozca internamente la estructura de los ficheros, si es capaz de manejarlos eficientemente gracias al uso de estas extensiones. Esta es la aproximación de los sistemas operativos de Microsoft.

Unix y sus variantes (Linux) sin embargo, optan por la no utilización de extensiones, lo que implica que el usuario es el único encargado de saber lo que se puede realizar o no con un fichero dado.

6.1. Estructuras de Directorios.

Veamos el siguiente ejemplo: Imaginemos un bufete de abogados que dispone de una ingente cantidad de información en papel: casos judiciales, precedentes, historiales de abogados, historiales de clientes, nóminas, cartas recibidas, copias de cartas enviadas, facturas del alquiler del local, albaranes de compra de lapiceros, procedimientos, etc.

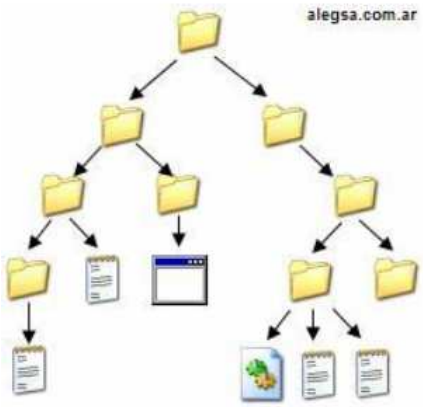
Ahora supongamos que todos estos documentos se almacenan en una enorme montaña de papel en el centro de una habitación: la locura está garantizada. Obviamente el bufete debe disponer de un armario de archivadores, la información se podrá almacenar de forma lógica para poder acceder a ella rápidamente cuando sea necesario.

Lo mismo ocurre en un sistema de ficheros informático: conviene guardar la información (los ficheros) en archivadores. Los archivadores serán lo que llamaremos directorios, un tipo especial de ficheros donde se almacena información relativa a otros ficheros.

Así, en un directorio se almacenarán ficheros relacionados entre sí, y ficheros totalmente independientes irán alojados en distintos directorios. Evidentemente, esta organización es puramente lógica: todos los ficheros estarán almacenados físicamente en el mismo lugar.

Hay varias formas de organizar los directorios sobre un disco:

- Directorio de un nivel. En este tipo de organización solo se permite un nivel de directorio.
- Directorio de dos niveles. En este tipo de organización, un directorio puede incluir dentro otro directorio, pero esté ya no puede incluir otro más.
- Directorio con estructura arborescente. Prácticamente no tiene limitaciones. Un directorio puede incluir otros directorios, sin importar su número, y estos nuevos directorios pueden contener otros directorios.



De esta forma, cada usuario puede crear sus propios directorios para organizar sus ficheros a su gusto. Un ejemplo de estructura de directorios de esta forma es la considerada por los sistemas de ficheros de Unix, MS-DOS, Windows, etc.

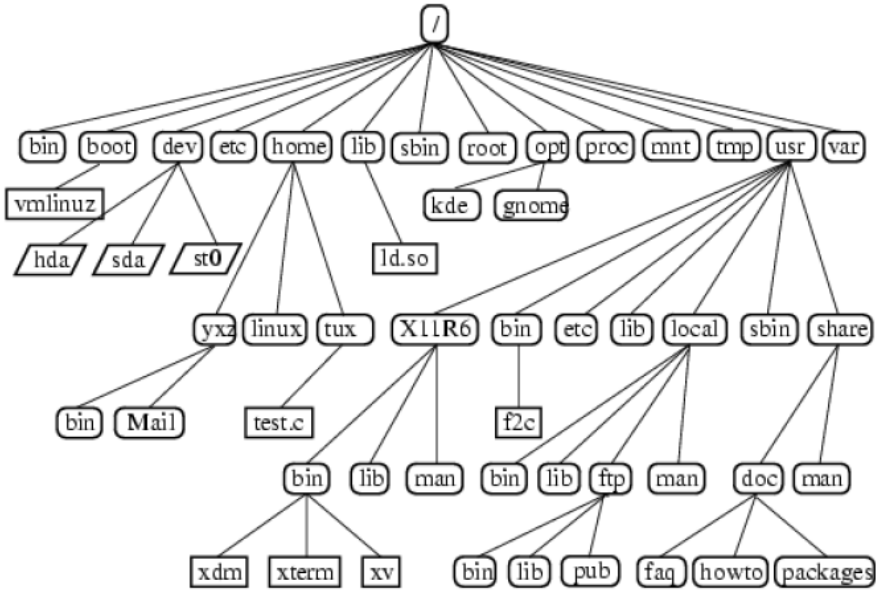
El árbol es de raíz única, de modo que cada fichero tiene un único nombre de ruta de acceso. El nombre de ruta de acceso en un directorio de esta forma es la concatenación de los nombres de directorio y subdirectorios desde el directorio raíz hasta el directorio donde se encuentra alojado el fichero a través del camino único, culminando con el propio nombre del fichero dentro del directorio.

Un directorio (o subdirectorio) contiene a su vez ficheros y/o subdirectorios, y todos los directorios poseen el mismo formato interno. Las entradas del directorio indican si el objeto referenciado es un fichero o un subdirectorio.

Se define directorio padre de un fichero o subdirectorio como el directorio en el que se encuentra su entrada de referencia Cada fichero o directorio (a excepción del directorio raíz) posee un único directorio padre. El directorio padre suele ser referenciado por los sistemas operativos con punto punto (..).

Se define directorio hijo de un directorio como el directorio que tiene por padre al primero. Un directorio puede contener múltiples directorios hijos, y cada directorio (a excepción de la raíz) es hijo de algún otro.

Se define directorio actual como aquel en el que trabaja el usuario por defecto. Suele ser referenciado por los sistemas operativos con un punto (.). Cuando el usuario hace referencia a un fichero por nombre (no por nombre de ruta de acceso), el sistema inicia la búsqueda siempre en el directorio actual. Si no lo encuentra, comienza a recorrer el camino de búsqueda hasta dar con él. El usuario puede referirse a un fichero también por su nombre de ruta de acceso, en cuyo caso no se da lugar a emplear el camino de búsqueda. El usuario también puede cambiar su directorio actual, especificando un nombre de directorio.



En este caso, los nombres de ruta de acceso pueden adoptar dos formas alternativas. La primera es la del nombre de ruta absoluto, la que hemos definido, por la cual se nombra a cada fichero con respecto al directorio raíz.

Por ejemplo, un nombre de ruta absoluto válido es C:\Documentos\Jose\Privado\Carta.txt. Normalmente, se denota el directorio raíz por medio de un símbolo especial dependiente de cada sistema de ficheros, aunque los más habituales son los símbolos '/' y '\'. En una estructura de directorio de este tipo, el nombre de ruta de acceso absoluto de un fichero o directorio debe ser único.

La segunda opción es la del nombre de ruta relativo. En este caso, se nombra al fichero con respecto al directorio actual. Para esta labor se definen en cada directorio dos entradas de directorio especiales: "." (Un punto) que representa al propio directorio y ".." (Dos puntos), que representa a su directorio padre. Así, se puede considerar un camino único desde el directorio actual hasta cualquier fichero o directorio del sistema. Un ejemplo de ruta relativa valida podría ser por ejemplo ..\Privado\Carta.txt o Jose\Privado\Carta.txt.

Los ficheros se van almacenando en el dispositivo, y por cada uno de ellos se apunta una entrada de directorio, donde almacenamos información sobre el tipo de fichero, nombre y demás. Hay que notar que por cada fichero se almacena por un lado el propio fichero, los datos, y por otro lado se almacena esta entrada de directorio.

Pero, ¿qué se almacena realmente en una entrada de directorio? La siguiente tabla muestra algunas de estas informaciones, aunque en un sistema de ficheros concreto pueden no estar todas las que son ni ser todas las que están:

Nombre del fichero	El nombre simbólico del fichero.
Tipo de fichero	Para aquellos sistemas que contemplan diferentes tipos.
Ubicación	Un puntero al dispositivo y a la posición del fichero en el dispositivo.
Tamaño	Tamaño actual del fichero (en bytes, palabras o bloques) y el máximo tamaño permitido.
Posición actual	Un puntero a la posición actual de lectura o escritura sobre el fichero. Su lugar exacto en el dispositivo.
Protección	Información referente a los permisos de acceso al fichero.
Contador de uso	Número de procesos que están utilizando simultáneamente el fichero.
Hora y fecha	De creación, último acceso, etc.

Identificación	Del proceso creador del fichero, del último que accedió al fichero, en qué forma lo hizo, etc.

Según el sistema concreto, una entrada de directorio puede ocupar desde una decena hasta varios miles de bytes. Por esta razón, la estructura de directorios suele almacenarse en el propio dispositivo que se está organizando, como un fichero especial.

La forma de almacenar los directorios en el dispositivo (estructura de datos del directorio) varía también de un sistema a otro. La forma más sencilla es la de una lista lineal, de forma que cada entrada se encuentra inmediatamente a continuación de la precedente. Esto sin embargo es muy caro en ejecución, pues la búsqueda de una entrada concreta suele exigir pasar por todas las anteriores (búsqueda lineal). Hay otros problemas, como el de qué hacer cuando se elimine una entrada (algunos sistemas la marcan como borrada para su posterior utilización, otros la incluyen en una lista de entradas libres).

Otra opción es mantener una lista lineal ordenada, pero exige el mantener permanentemente ordenada la lista -lo que complica excesivamente las operaciones de borrado y creación de entradas- y la complicación del algoritmo de búsqueda. Puede pensarse entonces en emplear un árbol binario de búsqueda, que simplifica las operaciones antes mencionadas (y complica el algoritmo de búsqueda).

6.2. Métodos de asignación.

Nos apartamos en este punto de la definición de fichero como tipo abstracto de datos y pasamos a considerar un aspecto bastante crítico: la forma de ubicar los ficheros físicamente sobre el disco o, dicho de otro modo, los distintos métodos existentes para asignar espacio a cada fichero dentro del disco. Por supuesto, éste es un aspecto totalmente transparente al usuario: al usuario no le interesa (o no tiene porqué interesarle) en absoluto la forma en que se almacenan físicamente los ficheros. Esto sólo interesa al desarrollador del sistema operativo o al programador de sistemas que necesita estar en contacto con las peculiaridades físicas del dispositivo.

Parece totalmente lógico que la asignación del espacio a los ficheros deba hacerse de modo que ésta sea tan eficiente como sea posible y que las operaciones a realizar sobre los ficheros sean rápidas. Contemplaremos tres métodos de asignación diferentes:

- Asignación contigua
- Asignación enlazada
- Asignación indexada

Algunos sistemas soportan las tres, pero lo más normal es que un sistema determinado sólo soporte uno.

Para empezar, es conveniente conocer cómo se administra el espacio libre del disco.

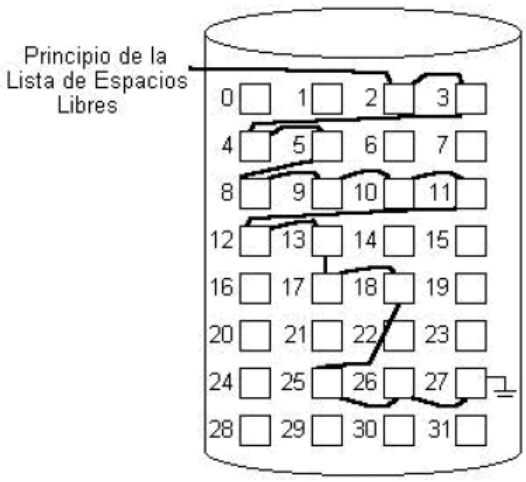
ADMINISTRACIÓN DEL ESPACIO LIBRE

En un sistema informático, los ficheros se crean y se destruyen con frecuencia. Debido a que el espacio de disco no es ilimitado, se hace necesario reutilizar el espacio ocupado por ficheros que han sido borrados para almacenar nuevos ficheros.

El sistema operativo debe mantener, pues, una lista de bloques (clusters) libres (no asignados a ningún fichero).

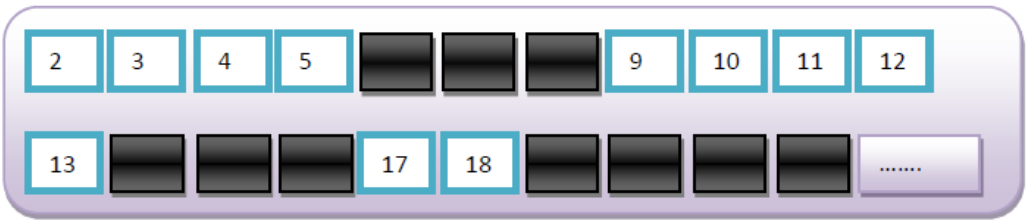
A la hora de crear un fichero, el sistema operativo examina esta lista en busca de bloques libres, los asigna y elimina dichos bloques de la lista. Cuando se borra un fichero de forma efectiva, los bloques que éste ocupaba se añaden a la lista de bloques libres.

Dicha lista de bloques libres se puede implementar de una forma similar a la que ilustra la figura. El sistema sólo mantiene un puntero al primer bloque libre (*), este primer bloque libre apunta al siguiente, y así sucesivamente hasta llegar al último bloque libre. (Lista encadenada).



Una forma similar de no perder de vista a los bloques libres, sin el engorro (en cuanto a espacio ocupado) que supone el mantener una lista en memoria, es la de implementar un mapa de bits (también llamado vector de bits o, en inglés, bitmap). En este esquema, cada bloque del disco se representa por medio de un bit, que estará puesto a un valor lógico concreto si el bloque está asignado a algún fichero y a su complementario cuando el bloque esté libre. En el disco del ejemplo mostrado en la figura anterior, estaban libres los bloques 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26 y 27; el mapa de bits de espacios libres correspondiente sería (suponiendo que el valor lógico 1 indica los bloques ocupados):

11000011000000111001111111000111111



Ahora guardaríamos este bitmap en nuestro dispositivo de almacenamiento, y solo ocuparía uno o dos bloques normalmente.

La lista de bloques libres, además de ocupar mucho más que un bitmap, exige leer cada bloque libre para obtener la posición del siguiente, lo cual requiere una cantidad de tiempo considerable de E/S.

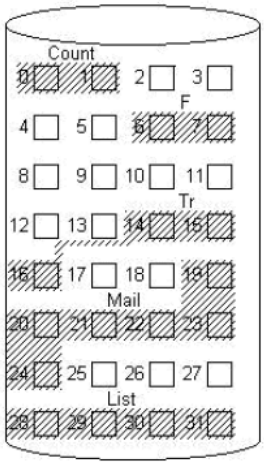
Existen otras alternativas, como son la de mantener una lista de bloques libres ligeramente modificada: se almacenan en el primer bloque libre las referencias a n bloques libres; de éstos, únicamente n - 1 están realmente libres: el último almacena otra n referencias a otros tantos bloques libres. Esta filosofía permite hallar un gran número de bloques libres de forma muy rápida.

Otra alternativa podría consistir en aprovechar que usualmente se asignan o liberan varios bloques contiguos de forma simultánea (sobre todo si se emplea la asignación contigua). Bastaría con almacenar la dirección del primer bloque liberado y el número de bloques libres que le siguen en un contador.

ASIGNACIÓN CONTIGUA

El método de asignación contigua funciona de forma que cada fichero ocupe un conjunto de bloques consecutivos en el disco. Como se dijo en apartados anteriores, cada bloque del disco posee una dirección que confiere una organización lineal al conjunto de bloques (los bloques están seguidos uno detrás de otro). De esta forma el acceso al bloque i+1 desde el bloque i no requiere normalmente movimiento alguno de la cabeza de lectura/escritura y, cuando sí lo requiere, se trata sólo de saltar a la pista siguiente.

Con la asignación contigua, la situación de un fichero queda perfectamente determinada por medio de la dirección de su primer bloque y su longitud. Sabiendo que un fichero tiene una longitud de n bloques y que comienza en el bloque b, entonces se sabe que el fichero ocupa los bloques b, b+1, b+2,... hasta b+n-1. Así pues, la única información relativa a la posición del fichero que es necesario mantener en la entrada del directorio consiste en la par dirección del primer bloque, longitud. En la figura podemos ver un ejemplo de este tipo de asignación.



DIRECTORIO

Fichero	Principio	Longitud
Count	0	2
Tr	14	3
Mail	19	6
List	28	4
F	6	2

Como puede verse, el acceso a un fichero cuyo espacio ha sido asignado de forma contigua es bastante sencillo.

El verdadero problema aparece a la hora de encontrar espacio para un nuevo fichero. Si el fichero a crear debe tener n bloques de longitud, habrá que localizar n bloques consecutivos en la lista de bloques libres. Así, en la figura anterior sería imposible almacenar un fichero que tuviera un tamaño de 7 bloques, aunque en realidad tenemos libres 15 bloques. (Fragmentación externa).

Vimos anteriormente que la fragmentación interna era algo inevitable (el espacio que se pierde en un cluster o bloque), sin embargo, la fragmentación externa sí se puede resolver. Una solución se llama compactación o desfragmentación. Cuando aparece una pérdida significativa de espacio en disco por culpa de la fragmentación externa, el usuario (o el propio sistema de forma automática) puede lanzar una rutina de empaquetamiento que agrupe todos los segmentos ocupados generando un único gran hueco. El único coste es el tiempo empleado en llevar a cabo esta compactación.

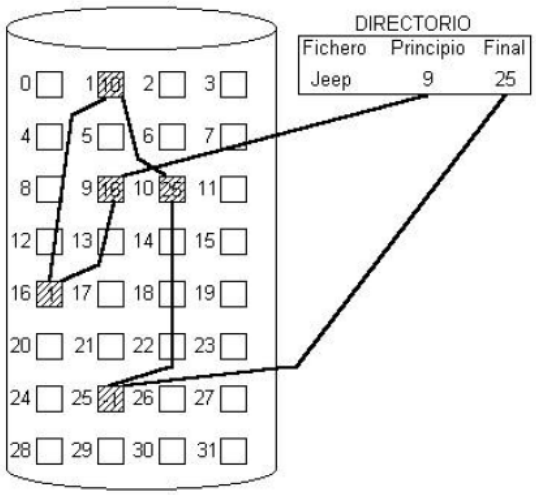
La asignación contigua presenta algunos otros problemas: esta estrategia parte del hecho de conocer el espacio máximo que ocupará un fichero en el instante mismo de su creación, algo que no siempre (mejor dicho, casi nunca) ocurre. Se puede exigir al usuario (humano o proceso) que señale explícitamente la cantidad de espacio requerido para un fichero dado en el momento de crearlo. Sin embargo, si este espacio resultara ser demasiado pequeño, el fichero no podría ampliarse en el futuro, y concluiríamos con un mensaje de error o una solicitud de mayor espacio. Normalmente, el usuario sobreestimará la cantidad de espacio necesaria, con lo que se derivará en un despilfarro de espacio muy importante.

Otra solución podría estar en localizar un hueco mayor cuando el hueco actual se quede pequeño, copiar todo el fichero al nuevo hueco y liberar el anterior. Sin embargo, esto ralentiza excesivamente al sistema. La pre asignación del espacio necesario puede no ser una solución válida, aunque se conozca exactamente el tamaño que alcanzará el fichero. Puede que el fichero vaya creciendo lentamente en el tiempo hacia su tamaño total, pero el espacio asignado no se utilizaría durante todo ese tiempo.

Concluyendo, la asignación contigua es muy eficiente a la hora de acceder a los ficheros, (de hecho, es la asignación más rápida en este aspecto) pero es excesivamente engorrosa a la hora de asignar el espacio a nuevos ficheros.

ASIGNACIÓN ENLAZADA

Para resolver los problemas de la asignación contigua, había que crear una estrategia de asignación que permitiera almacenar los ficheros de forma no continua, una forma de conseguir esto es mediante la asignación enlazada. Siguiendo este esquema, cada fichero no es más que una lista enlazada de bloques, que pueden encontrarse en cualquier lugar del disco. La entrada del directorio posee únicamente un puntero al primer bloque y un puntero al último. Cada bloque, a su vez, contendrá un puntero al siguiente bloque. Si suponemos bloques de 4 KB (4.096 Bytes), y usamos 4 Bytes (hasta 4.294.967.296 bloques podríamos usar) para almacenar cada puntero, cada bloque tendría un espacio útil de 4.092 Bytes.



En la figura siguiente podemos ver un fichero almacenado siguiendo esta estrategia. El fichero ocupa 5 bloques que son por este orden el 9, 16, 1, 10 y 25:

El proceso de creación de un fichero es muy fácil: sencillamente se crea una nueva entrada en el directorio dando a sus punteros de principio y final el valor null (valor nulo), indicando que el fichero está vacío.

El proceso de escritura simplemente escribe sobre un bloque libre, eliminándolo de la lista de bloques libres. Si se necesita otro bloque, se busca otro en la lista de bloques libres, se elimina a éste de la lista y se enlaza al anterior, actualizando el puntero al último de la entrada del directorio. La lectura del fichero únicamente consiste en ir siguiendo la cadena de punteros bloque a bloque. En el último bloque podemos colocar una marca en el puntero indicando que es el último.

BLOQUE 9	16

BLOQUE 16	1

BLOQUE 1	10

BLOQUE 10	25

BLOQUE 25	*

Vemos como la asignación enlazada aprovecha al máximo el tamaño de nuestro volumen, pues cualquier bloque libre es susceptible de ser utilizado en cualquier momento.

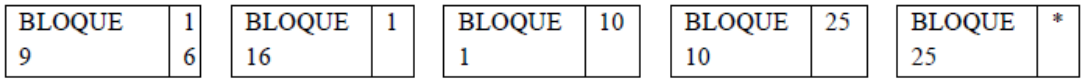
Tampoco es preciso conocer el tamaño de un fichero en el momento de su creación: el fichero puede crecer mientras existan bloques libres en el disco. Por este motivo no es indispensable compactar el disco como ocurría en la asignación continua, aunque si conveniente de vez en cuando desfragmentarlo para agilizar las operaciones de E/S.

Por supuesto, no todo va a ser ventajas. La asignación enlazada sólo es eficiente cuando se acceda a los ficheros de forma secuencial, pues el acceso al bloque i de un fichero exige recorrerlo desde el principio siguiendo la cadena de punteros hasta llegar al bloque deseado.

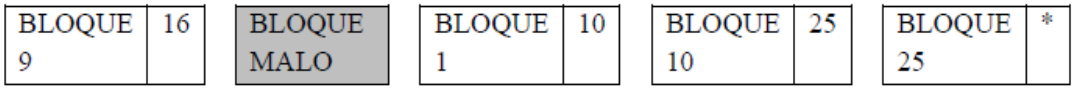
Puesto que no podemos acceder directamente a un bloque sin haber leído los anteriores, no se puede soportar el acceso directo a ficheros siguiendo esta filosofía. (No podemos empezar a leer un fichero por cualquier punto, siempre hay que empezar desde el principio).

Otra desventaja reside en el espacio desaprovechado para almacenar los punteros. En el ejemplo que vimos antes (4 KB por bloque y punteros de 4 Bytes) sólo se desperdicia un 0.09%, lo cual es muy poco, pero aun así es un 0.09% más del sitio que desperdicia la asignación contigua.

El mayor problema, y el más grave subyace en el aspecto de la fiabilidad del sistema: si, por cualquier causa, se dañara un único puntero en un bloque asignado a un fichero, el resto del fichero sería ilocalizable, y cabría la posibilidad de acceder a bloques no asignados o, peor aún, a bloques asignados a otros ficheros.

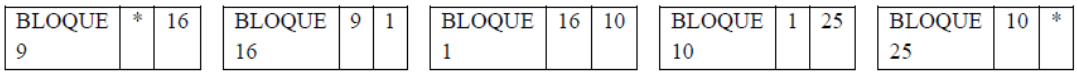


Imaginemos que, en el ejemplo anterior, que vemos arriba, se pierde un bloque del disco duro, en este caso el 16 por ejemplo, lo que quedaría:



De nuestro fichero, solo sería posible leer el primer bloque (el 9) ya que a partir de ahí no sabríamos a donde saltar. De esta manera, el fallo de un solo bloque de nuestro volumen de datos, podría hacer que se perdieran ficheros gigantescos. Y lo que es aún peor, el resto de bloques del fichero no quedarían marcados como libres, sino que sería basura que se quedaría en nuestro volumen para siempre.

La solución a este problema puede pasar por emplear una lista doblemente enlazada. Esta lista consiste en utilizar dos punteros en cada bloque, uno que apuntaría al anterior bloque, y otro que apuntaría al siguiente bloque. De esta manera podríamos leer un fichero desde el principio hasta el final, o desde el final hasta el principio. El ejemplo anterior quedaría así:



Puesto que en la entrada de directorio viene marcado el primer bloque y el ultimo, podríamos empezar a leer desde el final, así que aunque se estropeara el bloque 16 como ocurría antes, solo perderíamos un bloque del fichero realmente, ya que podríamos realizar una lectura hacia delante de 1 bloque, y un lectura hacia atrás de los otros 3 bloques del fichero.

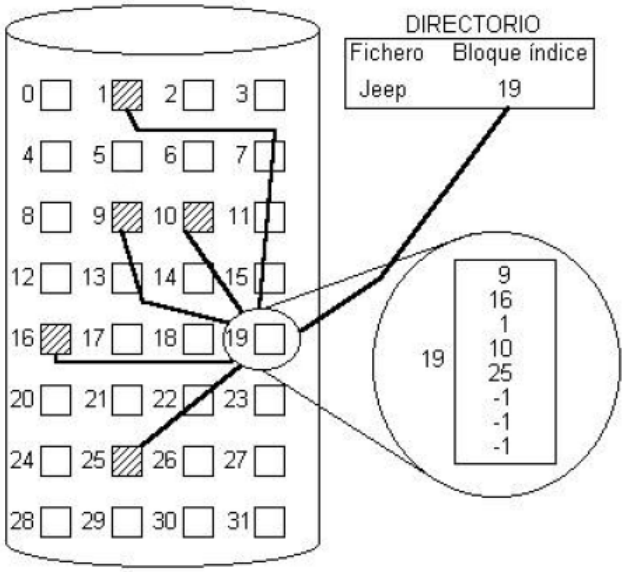
La desventaja de esta lista doblemente enlazada, es que se duplica el espacio desaprovechado en el volumen de datos, que pasa del 0,09% al 0,18% que ya es mucho.

ASIGNACIÓN INDEXADA

Hemos visto hace un momento cómo la asignación enlazada resolvía los problemas de la fragmentación externa grave y de la declaración del tamaño máximo del fichero en el momento de su creación. Sin embargo, vimos también que no daba soporte al acceso directo por estar todos los bloques del fichero (y sus punteros) dispersos por todo el disco.

La asignación indexada viene a resolver estos problemas reuniendo a todos los punteros en un mismo lugar: el bloque índice.

A cada fichero le corresponde su propio bloque índice, que no es más que una tabla de direcciones de bloques, donde la entrada i apunta al bloque i del fichero. La entrada de un fichero en el directorio sólo necesita mantener la dirección del bloque índice para localizar todos y cada uno de sus bloques, como puede verse en la figura siguiente:



Ahora sí se permite el acceso directo, pues el acceso al bloque i sólo exige emplear la i-esima entrada del bloque índice. El proceso de creación de un fichero implica inicializar todos los punteros de su bloque índice a null para indicar que éste está vacío. (-1 lo consideramos valor Null), y a continuación ir apuntando los bloques que va usando.

La escritura del bloque i-esimo por primera vez consiste en eliminar un bloque cualquiera de la lista de bloques libres y poner su dirección en la entrada i-esima del bloque índice.

Como se ve, la asignación indexada soporta el acceso directo sin provocar fragmentación externa grave. Cualquier bloque libre en cualquier lugar del disco puede satisfacer una solicitud de espacio.

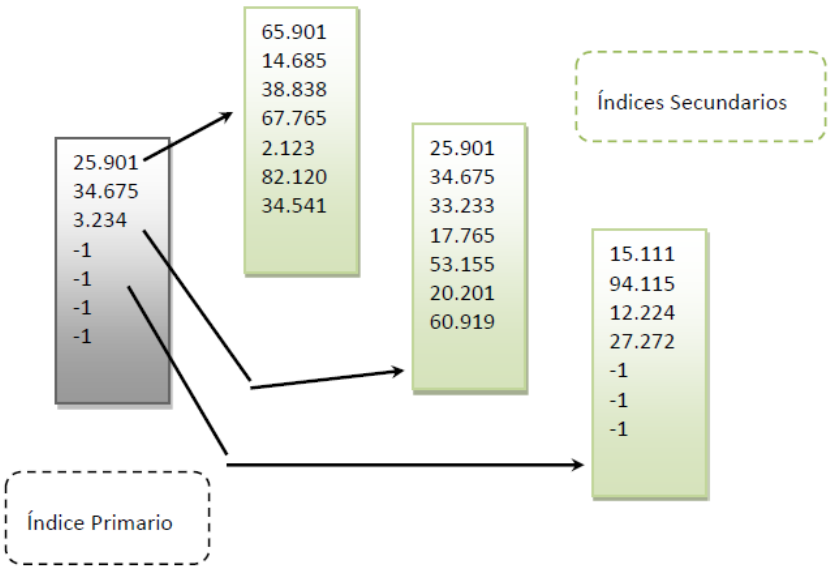
Sin embargo, también hay desventajas: los bloques índices son también bloques de disco, y dejan de estar disponibles para almacenar datos, así que perdemos algo de la capacidad del volumen. Como la mayoría de los ficheros existentes en un sistema son pequeños, el despilfarro puede ser bastante considerable, pues se exige reservar todo un bloque como índice para cada fichero, aunque éste sólo ocupe uno o dos bloques (uno o dos punteros efectivos dentro del índice y el resto conteniendo el valor null).

Si usamos por ejemplo un cluster de 32 KB, y almacenamos un fichero de 10 KB, estaremos usando realmente 64 KB, 32 para almacenar el fichero y 32 para almacenar su bloque índice.

Y supongamos ahora un fichero realmente grande, es tan grande que su lista de bloques usados no cabe en un único bloque índice. Como un bloque índice es también un bloque de disco, podríamos enlazar varios de estos bloques índices para conseguir uno mayor. Por ejemplo, un bloque índice podría contener un pequeño encabezamiento con el nombre de fichero, y las cien primeras direcciones de bloques de disco. La siguiente dirección (la última palabra en el bloque índice) es null (para un fichero pequeño) o bien un puntero a un segundo bloque índice. En el caso de un fichero muy muy grande, este segundo bloque índice podría apuntar al final a un tercero, etc.

Esta filosofía emplea una asignación enlazada para los bloques índices, lo que podría no resultar conveniente en términos de acceso directo. Como alternativa, podríamos pensar en emplear una estructura arborescente (de árbol binario) en dos o tres niveles. Un fichero pequeño sólo emplearía direcciones del primer nivel. A medida que los ficheros crecen puede acudirse a los niveles segundo y tercero.

El acceso a un bloque de datos exige acceder al puntero del primer nivel contenido en el bloque índice primario, el cual apunta a su vez a otro bloque índice secundario, etc. Por ejemplo, si podemos tener 256 punteros en un bloque índice, entonces dos niveles de índice nos permiten referenciar hasta 65.536 bloques de datos (256 x 256).



En este ejemplo de arriba vemos un sistema de asignación indexada de 2 niveles. Cada bloque índice puede almacenar 7 punteros, de modo que podemos asignar un máximo de 49 (7 x 7) bloques a un fichero. Podrimos crear un 3º nivel, haciendo que los punteros de los índices secundarios apuntaran a un 3º nivel de bloques índice, con lo que podríamos asignar un máximo de 343 (7 x 7 x 7) bloques a un fichero.

Otra alternativa consiste en mantener los primeros punteros del bloque índice, digamos unos 15, en el directorio de dispositivo. Si un fichero requiere de más de 15 bloques, un décimo sexto puntero apunta a una lista de bloques índice. De esta manera, los ficheros pequeños no precisan de un bloque índice.

6.3. Sistema de archivos.

El sistema de archivos o sistema de ficheros es el componente del sistema operativo encargado de administrar y facilitar el uso de las memorias secundarias, independientemente del medio sobre el que se implementen.

Sus principales funciones son la asignación de espacio a los archivos, la administración del espacio libre y del acceso a los datos resguardados. Estructuran la información guardada en un dispositivo de almacenamiento de datos o unidad de almacenamiento (normalmente un disco duro de una computadora), que luego será representada ya sea textual o gráficamente utilizando un gestor de archivos.

La mayoría de los sistemas operativos manejan su propio sistema de archivos.

6.3.1.NTFS.

NTFS (New Technology File System) es un sistema de archivos diseñado por y para sistemas Microsoft. Debido a que su especificación es un secreto tiene un soporte muy limitado en sistemas operativos no-Microsoft

ESTRUCTURA

SECTOR DE ARRANQUE (MBR)	MFT (MASTER FILE TABLE)	AREA DE DATOS	AREA DE METADATOS	AREA DE DATOS	COPIA DEL SECTOR DE ARRANQUE

- Todos los ficheros que están contenidos en el sistema de ficheros tiene una entrada en el MFT.
- En casos de ficheros pequeños y directorios que no excedan el límite del tamaño del registro MFT están completamente contenidos en el MFT.
- La sección de los datos almacena el resto de los datos de los ficheros.
- Existen réplicas de las partes más sensibles en caso de pérdida o error.

6.3.2.Sistema de archivos Linux.

El Linux existen varios sistemas de archivos, el sistema actual más usado es ext4, pero la estructura básica es común.

Estructura básica

SECTOR DE ARRANQUE (MBR)	SUPERBLOQUE	TABLA DE INODOS	ZONA DE DATOS

- El superbloque contiene datos sobre toda la partición, pero existen varias copias distribuidas por la partición.
- En la tabla de inodos almacena información sobre que inodos están libres y cuales ocupados.
- Existe un inodo por cada archivo existente en el sistema
- La tabla de inodos contiene todos los inodos de la zona de datos.

7. Gestion de entradas/salidas.

Una de las funciones más importantes y complejas que realiza el sistema operativo es la destinada a manejar los diferentes periféricos existentes ya que debe enviar instrucciones y datos a los dispositivos, detectar interrupciones y controlar errores.

Se puede definir una operación de E/S, como el intercambio de información entre la memoria principal o la CPU y los dispositivos E/S.

Existen multitud de dispositivos distintos entre sí, tanto en su funcionamiento como en sus características físicas y mecánicas, y el sistema operativo debe comunicarse con todos ellos y gestionarlos de similar para facilitar la comunicación con el usuario.

Algunos de los problemas entre el Sistema Operativo y los Dispositivos de entrada salida son:

- Cada dispositivo tiene diferentes funciones.
- Los dispositivos dependen de núcleo de sistema operativo.
- Existe diferentes velocidades entre los dispositivos tanto como de entrada, salida y bloque.
- El sistema operativo trata de buscar la mejor manera de realizar la función de entrada y salida.

Como cada dispositivo es distinto, el sistema operativo necesita instalar un software específico que controla ese hardware. Ese software se llama driver o controladora.

Los dispositivos de entrada/salida se pueden dividir según la forma de trabajar, y estos son:

- Dispositivos orientados a bloques: trabajan con bloques de tamaño fijo y tienen la propiedad de direccionar, es decir, se puede escribir o leer de cualquier posición. Ejemplos de estos discos duros, la memoria o los discos ópticos.
- Dispositivos orientados a caracteres, trabajan con secuencias de bytes sin importar su longitud ni ninguna agrupación en especial. No son dispositivos direccionables. Ejemplos de estos el teclado que no disponen de memoria, ni de procesador.

8. Instalación de sistema operativo.

8.1. Preparar el equipo para arrancar desde unidades externas.

Para instalar un sistema operativo será necesario poder elegir, y elegir el sistema externo, que contendrá un sistema de arranque. Si no es configurado para ello habrá que modificar la configuración de la BIOS, para escoger la unidad externa como primer dispositivo para el arranque.

Esta operación depende del modelo de placa base/madre del equipo, por lo que de ser posible consultaremos la documentación del fabricante.

Normalmente, para acceder a la modificación de la configuración de BIOS, hay que pulsar la tecla “Suprimir”, “F2” en los primeros segundos del POST (comprobación del sistema en el encendido).

Luego, en la configuración avanzada, debemos cambiar el parámetro BOOT para que el primer dispositivo sea la unidad externa. Esta operación difiere mucho entre distintos ordenadores. Manipular otros parámetros de la BIOS puede dejar al ordenador inservible. Solicite ayuda a algún experto si no sabe qué hacer. Seleccione la opción de salir sin salvar los cambios (EXIT Without update).

Finalmente, hay que seleccionar la opción salvar cambios y salir (normalmente pulsando F10).

8.2. Preparación del Disco Duro.

Esta fase consiste en crear las particiones del tipo necesario para que nuestro S.O. pueda instalarse.

En Windows los tipos de particiones que se emplean son FAT32 (Windows 95/98) y NTFS (Windows NT/2000, XP, Windows 7, Windows 8.1 y Windows 10). En Linux/UNIX, se aceptan muchos más tipos de particiones, siendo el sistema de ficheros más popular el EXT4.

Si queremos instalar un sistema operativo en un disco donde ya haya otro sistema operativo instalado, ES MUY IMPORTANTE HACER COPIA DE SEGURIDAD DE LOS DATOS IMPORTANTES ANTES DE PROSEGUIR LA INSTALACIÓN, ya que existe un alto riesgo de perderlos TODOS por un error durante el proceso. Una vez hecho esto, tendremos dos opciones:

Sustituir el sistema operativo anterior.

Instalarlo permitiendo su coexistencia y selección durante el periodo de arranque del ordenador.

Si elegimos la primera opción, suele ser buena idea borrar en el proceso de instalación las particiones antiguas y después crear las nuevas, realizando una comprobación completa de su estado para conocer si hay errores o defectos en el disco.

En el caso de querer hacer una instalación dual, habrá que conseguir espacio suficiente para instalar el nuevo sistema operativo, normalmente restándoselo a las particiones existentes anteriormente para el primer sistema. Esta delicada tarea, suele hacerse con herramientas software especial, como “Partition Magic” o la libre y gratuito bajo Linux QTParted o Gparted. En ambos casos, es muy recomendable realizar una sola modificación cada vez y llevarla a efecto, en lugar de programar varias encadenadas.

Los programas instaladores de Linux, suelen incorporar herramientas que permiten dicha modificación. No ocurre así en los de Windows, que solo permiten borrar antiguas y crear nuevas.

Suele ser muy interesante por motivos de seguridad, crear particiones independientes para guardar los datos de los usuarios (por ejemplo, una unidad D: en Windows, o directorio /home en Linux).

8.3. Planificación de la instalación.

Indistintamente de que estemos instalando una nueva red desde cero o estemos modificando una implantación que ya se encuentra en funcionamiento, debemos planificar tanto los objetivos como las necesidades que se derivan de ellos, dedicando especial atención a:

- Los recursos que deben estar disponibles.
- Estimación de tráfico originado (tanto en la comunicación de los clientes con los servidores como en la comunicación entre los propios servidores).
- Relación de elementos que reutilizaremos de la instalación anterior, si existen.
- Detalle de elementos que deben adquirirse, incluida su valoración.
- Estudio de la compatibilidad entre los diferentes componentes que formarán la infraestructura, tanto el hardware como el software, prestando especial atención a la disponibilidad de controladores de dispositivo para todos los elementos implicados y a la compatibilidad entre las diferentes aplicaciones, y de éstas con el Hardware que emplearemos.
- Trabajos de adecuación que deben afrontarse. En este punto y en el anterior, debemos tener en cuenta que un proyecto será más viable cuanto más contenga sus gastos sin menoscabar el servicio que ofrece. En realidad, ese es el principal objetivo de todo buen proyecto.
- Lugar en el que se ubicará cada elemento de la instalación. Puede ser adecuada la inclusión de un plano del edificio en el que aparezcan indicadas las diferentes localizaciones a las que se haga referencia.
- Configuración de los distintos dispositivos.

Las funciones que desempeñará cada elemento.

Por último, deberá realizarse una estimación de futuro, para intentar vislumbrar si el sistema deberá evolucionar en tamaño o en funcionalidad a corto o medio plazo. De esta forma, trataremos de evitar, en la medida de lo posible volver a modificar a corto o medio plazo la instalación que estamos realizando ahora.

Un aspecto general, que también deberá estar incluido es la relación de las tareas que serán desarrolladas por miembros de la empresa y cuáles serán encargadas a personal externo (sin olvidar, en ningún caso su valoración).

Es importante dedicar a la fase de elaboración del proyecto todo el tiempo necesario, para no precipitarnos en las decisiones.

A continuación, llevaremos a cabo la tarea siguiendo el proyecto anterior, prestando especial atención a la organización y documentando cada aspecto de la tarea realizada.

Al terminar, realizaremos una comprobación detallada de la instalación asegurándonos que cada elemento cumple sus funciones de forma satisfactoria y documentaremos los resultados obtenidos.

8.4. Requisitos previos.

Antes de la instalación se requiere unos requisitos mínimos. Los requisitos que se miran son principalmente, el procesador (la arquitectura de los registros y la velocidad), memoria RAM y el espacio en disco. A veces también los sistemas operativos tienen requerimientos de tarjeta gráfica. Se suelen especificar 2 términos los requisitos mínimos y los recomendados. Así, por ejemplo, un Windows 10, necesita:

Recurso	Requisitos mínimos	Requisitos recomendables
Procesador	1 gigahercio	> a 1 gigahercio
Memoria RAM	1 GB	2 GB
Espacio en disco	16 GB	20 GB

Para los sistemas actuales, existen distribuciones para arquitecturas de 32 y de 64 bits.

8.5. Ejecutar el programa de instalación.

Para ello, normalmente bastará con conectar la unidad externa con el software de arranque y seleccionar ese dispositivo. Debemos estar atentos a los primeros instantes para leer un posible mensaje de proceder a la instalación y aceptarlo. En caso contrario, bastará con esperar sin hacer nada.

Durante el proceso de instalación de la mayoría de los sistemas operativos, se configura y selecciona una serie de parámetros, como la hora del sistema, la región, el teclado o los parámetros de la red. Si nuestro equipo va a ser utilizado en una red local o en Internet, habremos de configurar adecuadamente el dispositivo de comunicaciones (normalmente la tarjeta de red). Para ello, necesitaremos obtener la información pertinente del administrador de la red o del proveedor de servicios de Internet que tengamos contratado en su caso.

Lo más común, (y por tanto la instalación por defecto) es que los equipos se configuren de modo que automáticamente consigan el ajuste necesario de la red desde otro equipo que los coordina a todos, mediante un protocolo denominado DHCP (Dinamic Host Control Protocol). Si es así, no necesitamos hacer nada más.

En caso contrario, deberemos obtener y anotar la información correspondiente para la tarjeta de red.

Durante la instalación se debe proporcionar el nombre y contraseña del usuario que será administrador del sistema. Todo sistema multiusuario que se precie, debe tener un responsable de su funcionamiento, mantenimiento y de otorgar permisos de uso del equipo y/o sus recursos a terceros. Es durante la fase de instalación durante la que se especifica la contraseña para el mismo. En los sistemas UNIX, el nombre del administrador es siempre “root”. En sistemas como Windows, UBUNTU o GuadaLinex, esta labor la lleva el primer usuario creado, hasta que se especifique lo contrario.

También durante la instalación se debe seleccionar los componentes software opcionales que queremos instalar. Muchas distribuciones de S.O., especialmente de la familia Linux, pueden contener software adicional que puede ser instalado durante la instalación del mismo. Es habitual que se nos pregunte por qué selección de programas recomendada o personalizada queremos instalar. Una vez hecho esto, comienza la copia de todos los ficheros necesarios desde los soportes de instalación al disco duro del equipo.

9. Sistemas operativos de Windows.

Microsoft ha mantenido 2 líneas de sistemas operativos, los sistemas operativos de hogar y los servidores. Ahora Microsoft está implantando una tercera línea, la de los dispositivos móviles. Así la evolución en los sistemas operativos de escritorio es:

- MS-DOS
- Windows 3.1
- Windows 95
- Windows 98
- Windows Me
- Windows XP.
- Windows Vista
- Windows 7.
- Windows 8.
- Windows 8.1.
- Windows 10.

La evolución de los sistemas profesionales es:

- Windows NT.
- Windows 2000.
- Windows 2003.
- Windows 2008.
- Windows 2012.
- Windows 2016.

A la familia para móviles se les llama Windows CE y la evolución es la siguiente:

- PocketPc 2002
- Windows Mobile 2003
- Windows Mobile 5.
- Windows Mobile 6.
- Windows Mobile 7.

Versiones Server y Desktop relacionadas			
Windows Server		Windows Desktop	
Lanza-miento	Versión	Lanza-miento	Versión
Feb-2000	Windows 2000 (NT 5.0)	Oct-2001	Windows XP (NT 5.1)
Abr-2003	Windows Server 2003 (NT 5.2)	Abr-2005	Windows XP x64 (NT 5.2)
Feb-2008	Windows Server 2008 (NT 6.0)	Ene-2007	Windows Vista (NT 6.0)
Sep-2009	Windows Server 2008 R2 (NT 6.1)	Oct-2009	Windows 7 (NT 6.1)
Sep-2012	Windows Server 2012 (NT 6.2)	Oct-2012	Windows 8 y Windows RT (NT 6.2)
Oct-2013	Windows Server 2012 R2 (NT 6.3)	Oct-2013	Windows 8.1 (NT 6.1)
Oct-2016	Windows Server 2016 (NT 10.0)	Jul-2015	Windows 10 (NT 10.0)
Nota: Windows RT es una edición de Windows 8 para dispositivos móviles que usan arquitectura ARMv7 de 32 bits.			

10. Distribuciones de Linux.

Una distribución Linux (coloquialmente llamada distro) es una distribución de software basada en el núcleo Linux que incluye determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios, dando así origen a ediciones domésticas, empresariales y para servidores. Por lo general están compuestas, total o mayoritariamente, de software libre, aunque a menudo incorporan aplicaciones o controladores propietarios.

En general, las distribuciones Linux pueden ser:

- Comerciales o no comerciales.
- Ser completamente libres o incluir software privativo.
- Diseñadas para uso en el hogar o en las empresas.
- Diseñadas para servidores, escritorios o dispositivos empotrados.
- Orientadas a usuarios regulares o usuarios avanzados.
- De uso general o para dispositivos altamente especializados, como un cortafuego, un enrutador o un cluster computacional.
- Diseñadas e incluso certificadas para un hardware o arquitectura específicos.
- Orientadas hacia grupos en específico, por ejemplo, a través de la internacionalización y localización del lenguaje, o por la inclusión de varios paquetes para la producción musical o para computación científica.
- Configuradas especialmente para ser más seguras, completas, portables o fáciles de usar.
- Soportadas bajo distintos tipos de hardware.

La diversidad de las distribuciones Linux es debido a cuestiones técnicas, de organización y de puntos de vista diferentes entre usuarios y proveedores. El modo de licenciamiento del software libre permite que cualquier usuario con los conocimientos e interés suficiente pueda adaptar o diseñar una distribución de acuerdo a sus necesidades.

Dentro de todas las distribuciones, existen 2 especialmente y sobre la que se basan el resto de distribuciones. Estas son Red Hat y Debian, que son los principales gestores de paquetes de software. Así para instalar software existen 3 maneras distintas:

- Descargarse (comprar), los paquetes compilarlos para su máquina e instalarlos.
- Descargarse los paquetes precompilados para la distribución que tenemos instalada. Así si la distribución es Red Hat o derivada de esta se instalan los paquetes .rpm y si la distribución es Debian o derivada de esta, se instalaran paquetes .deb.
- Usar un software para la gestión del software, que se encargara de buscar en los repositorios e instalar el software. Ejemplo de estos gestores son Yast, Synaptic, Adept, apt o dpkg.

Entre las distribuciones Linux más populares se incluyen:

- CentOS, una distribución creada a partir del mismo código del sistema Red Hat pero mantenida por una comunidad de desarrolladores voluntarios.
- Debian, una distribución mantenida por una red de desarrolladores voluntarios con un gran compromiso por los principios del software libre.
- Fedora, una distribución lanzada por Red Hat para la comunidad.
- Gentoo, una distribución orientada a usuarios avanzados, conocida por la similitud en su sistema de paquetes con el FreeBSD Ports, un sistema que automatiza la compilación de aplicaciones desde su código fuente.
- gOS, una distribución basada en Ubuntu para netbooks.
- Knoppix, la primera distribución live en correr completamente desde un medio extraíble. Está basada en Debian.
- Kubuntu, la versión en KDE de Ubuntu.

- Linux Mint, una popular distribución derivada de Ubuntu.
- Mandriva, mantenida por la compañía francesa del mismo nombre, es un sistema popular en Francia y Brasil. Está basada en Red Hat.
- openSUSE, originalmente basada en Slackware es patrocinada actualmente por la compañía Novell.
- PCLinuxOS, derivada de Mandriva, paso de ser un pequeño proyecto a una popular distribución con una gran comunidad de desarrolladores.
- Puppy Linux, versión para pc's antiguas o con pocos recursos que pesa 130 mb.
- Red Hat Enterprise Linux, derivada de Fedora, es mantenida y soportada comercialmente por Red Hat.
- Slackware, una de las primeras distribuciones Linux y la más antigua en funcionamiento. Fue fundada en 1993 y todavía operativa y actualizada.
- Slax, es un sistema Linux pequeño, moderno, rápido y portable orientado a la modularidad. Está basado en Slackware.
- Ubuntu, una popular distribución para escritorio basada en Debian y mantenida por Canonical.

11. Otros sistemas operativos.

Además de estas 2 familias, existen otros sistemas operativos, entre los que cabe destacar:

Mac OS X: es un sistema operativo desarrollado y comercializado por Apple Inc. que ha sido incluido en su gama de computadoras Macintosh desde 2002 Es el sucesor del Mac OS 9 (la versión final del Mac OS Classic), el sistema operativo de Apple desde 1984. Está basado en UNIX, y se construyó sobre las tecnologías desarrolladas en NeXT entre la segunda mitad de los 80's y finales de 1996, cuando Apple adquirió esta compañía. Desde la versión Mac OS X 10.5 Leopard para procesadores Intel.

Unix: Unix (registrado oficialmente como UNIX®) es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T.

Existen varias familias del sistema operativo UNIX, que han evolucionado de manera independiente a lo largo de los años. Cada familia se distingue no tanto por sus diferencias técnicas como por sus diferencias en propiedad intelectual. Se observa que todas las familias se han visto contaminadas, directa o indirectamente, por otras familias.

- AT&T: la familia que tuvo su origen en el UNIX de AT&T. Considerada la familia UNIX "pura" y original. Sus sistemas operativos más significativos son UNIX System III y UNIX System V.
- BSD: familia originada por el licenciamiento de UNIX a Berkely. BSD se reescribió para no incorporar propiedad intelectual originaria de AT&T en la versión 4. La primera implementación de los protocolos TCP/IP que dieron origen a Internet son la pila (stack) TCP/IP BSD.
- AIX: Esta familia surge por el licenciamiento de UNIX System III a IBM.
- Xenix: familia derivada de la adquisición de los derechos originales de AT&T primero por parte de Microsoft y de esta los vendió a SCO.

Solaris: es un sistema operativo de tipo Unix desarrollado desde 1992 inicialmente por Sun Microsystems y actualmente por Oracle Corporation como sucesor de SunOS. Es un sistema certificado oficialmente como versión de Unix. Funciona en arquitecturas SPARC y x86 para servidores y estaciones de trabajo. Existe una versión libre denominada openSolaris.

Otros: Además existen otros sistemas operativos parcialmente obsoletos como MS-DOS y OS/2.

12. Actualización del sistema operativo.

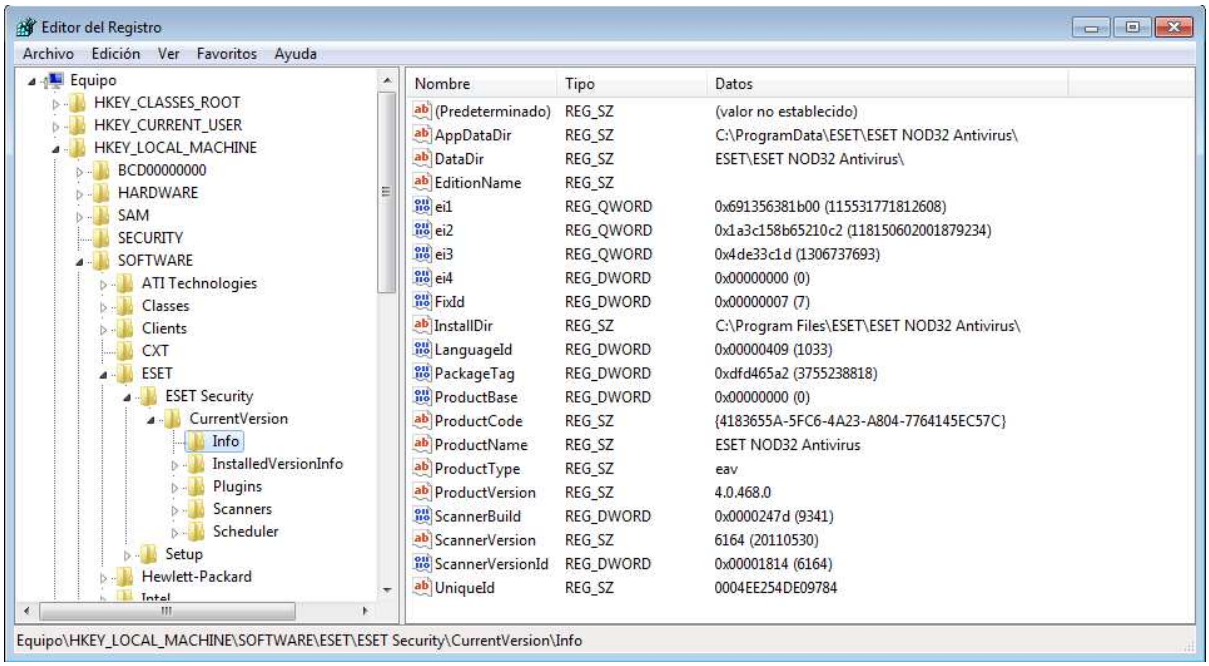
Quizás se pueda pensar que un sistema operativo es un producto terminado y que las actualizaciones, aunque interesantes, no son realmente importantes. De hecho, hay usuarios que ven la instalación de actualizaciones como una pérdida de tiempo. Sin embargo, estas son ideas muy alejadas de la realidad, ya que los sistemas operativos actuales son productos en continua evolución y, cuando están actualizados no sólo funcionan mejor, sino que serán más seguros. No hay que olvidar que algunas de las actualizaciones resuelven vulnerabilidades que han sido encontradas, bien por el equipo de desarrollo del producto, bien por los propios usuarios.

13. Registro de Windows.

El registro de Windows o registro del sistema es la base de datos que almacena las configuraciones y opciones del sistema operativo Microsoft Windows en sus versiones de 32 bits, 64 bits y Windows Mobile.

El registro de Windows contiene información y configuraciones de todo el hardware, software, usuarios, y preferencias del PC. Si un usuario hace cambios en las configuraciones del "Panel de control", en las asociaciones de ficheros, en las políticas del sistema o en el software instalado, los cambios se reflejan y almacenan en el registro.

El registro mantiene esta información en forma de árbol, estableciendo un orden por el cual deben acceder el sistema operativo u otros programas, como las preferencias de usuario (perfiles), hojas de ajustes para directorios e iconos de programas, enumeración de hardware instalado y los puertos usados.



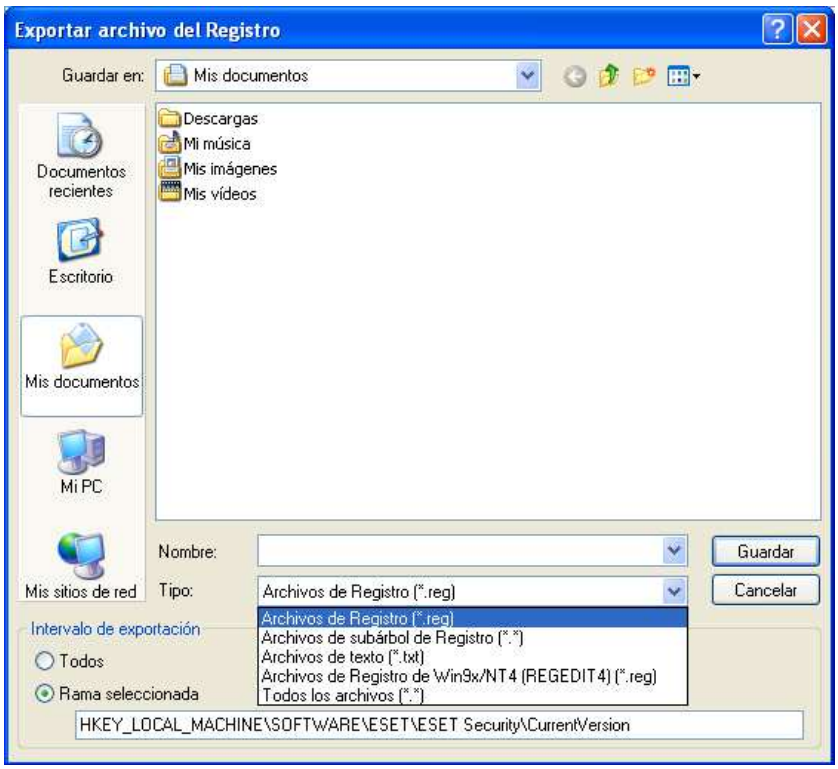
Para mostrar el registro de Windows, se utiliza la aplicación regedit.exe.
El registro es una base de datos jerarquizada donde con la siguiente estructura:
Categorías → Subcategorías → Claves → Subclaves → Entradas.

Los datos que contiene no pueden cancelarse, Si se comete un error no hay marcha atrás, así mucho cuidado a la hora de editar el Registro y nunca debe realizarse ningún cambio sin contar con una copia de seguridad actualizada del Registro. Con F8 se arranca con la última configuración válida conocida.

Cualquier cambio que se realice en el registro se aplicará de inmediato y no existe ninguna función deshacer. Es decir, no existe ninguna forma automática de dar marcha atrás.

Cualquier cosa que se borre en el registro se borra automáticamente.

El registro no espera a que se pulse "*Guardar*" ("*Save*") para guardar el registro en disco, los cambios se guardan automáticamente.



Una forma segura → "*Exportar"* previamente a un lugar seguro. Se puede copiar a un Archivo .reg (archivo de registro) o un archivo de subárbol de registro. La diferencia es la siguiente:

- Archivo .reg (archivo de registro)

Si se exporta, se añade una clave nueva al registro y luego se importa, la clave nueva seguirá existiendo.

Se guardan en modo texto son editables.

- Un archivo de subárbol realiza una copia total.

Si se exporta, se añade una clave nueva al registro y luego se importa, la nueva clave desaparece dejando el registro tal y como estaba a la hora de realizar la exportación.

Se guardan en formato especial, no pueden ser editados ni visualizados por métodos normales.

Windows presenta un comportamiento anómalo cuando se importan subárboles grandes, llegando a presentar problemas de memoria baja e incluso llegando a corromper partes del registro. Se recomienda no utilizar subárboles siempre que sea posible.

La estructura es en forma arborescente. En primer lugar, están las carpetas principales que cuelgan de la raíz del registro. Estas carpetas principales son:

En cada una de estas carpetas y en sus sucesivas subcarpetas o subdirectorios, se pueden crear tanto carpetas en la parte izquierda, como entradas en la parte derecha.

Clave	Descripción
HKEY_CURRENT_USER	Información sobre el usuario actual.
HKEY_USERS	Información sobre todos los usuarios.
HKEY_LOCAL_MACHINE	Información sobre el equipo.
HKEY_CLASSES_ROOT	Información sobre las extensiones y sus asociaciones.
HKEY_CURRENT_CONFIG	Información sobre la configuración actual que se esta usando.

Estas entradas pueden ser variables de distintos tipos de datos. Entre otros:

Tipos de datos	Descripción
REG_BINARY	Datos binarios en formato hexadecimal.
REG_DWORD	Datos de doble palabra.
REG_EXPAND_SZ	Cadena de texto de longitud variable.
REG_MULTI_SZ	Lista o array separados por espacios.
REG_SZ	Cadena de texto de longitud fija.
REG_FULL_RESOURCE_DESCRIPTOR	Tablas anidadas.

14. 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.

14.1. 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:

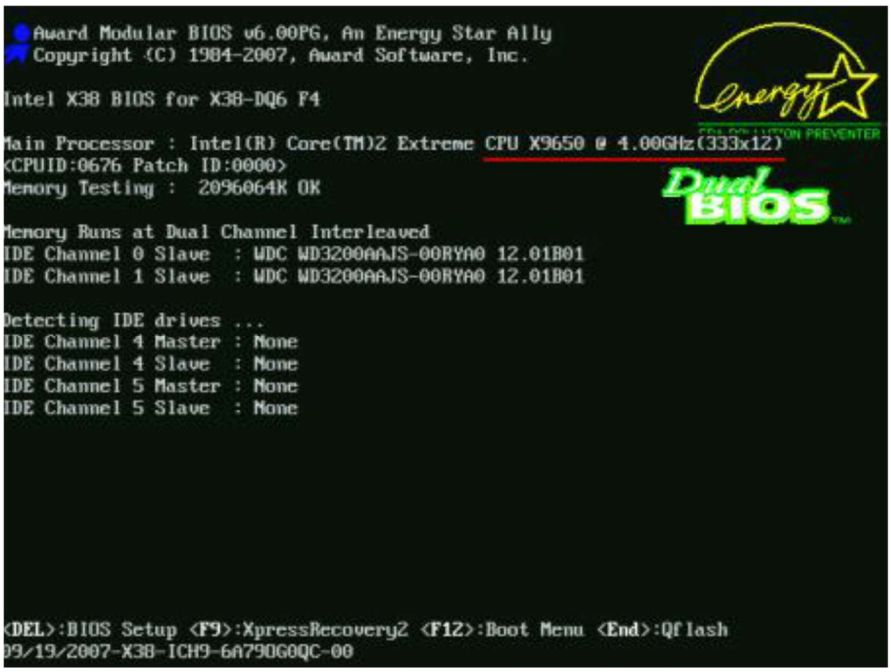
1. Cuando se da tensión a la fuente de alimentación y una vez que la alimentación se estabiliza, genera una señal denominada “Power Good” en uno de los cables que va de la fuente de alimentación a la placa base; esta señal es recibida en el juego de chips instalado en la referida placa, y a su vez generan 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:

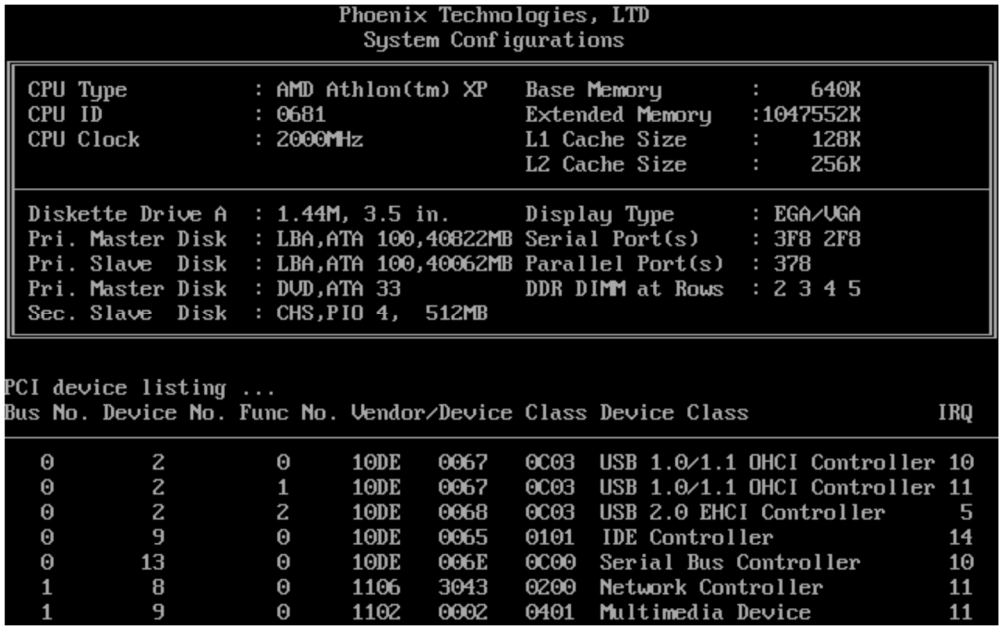
a) Comprobación del procesador
b) Varias comprobaciones sobre la memoria RAM
c) Inicializar los dispositivos de video y teclado. La comprobación del dispositivo de video incluye cargar y ejecuta la parte de BIOS incluida en el adaptador de video. La mayoría de las adaptadoras modernas muestran en pantalla 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).



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.



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.

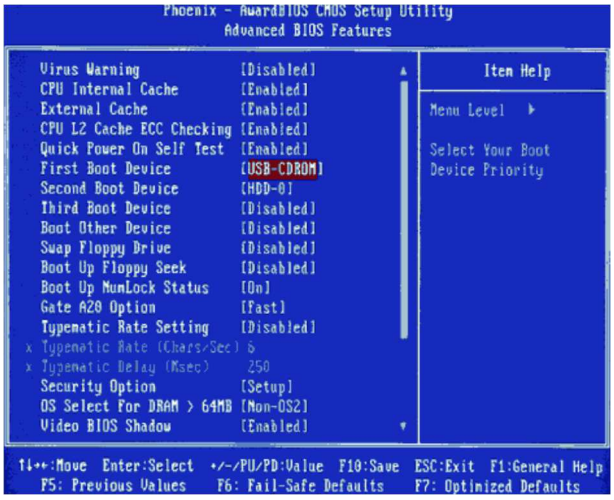
¿Pero dónde buscará el POST el programa a cargar? Y en caso de que existan varios sistemas operativos en varios soportes, ¿cuál de ellos será el elegido?

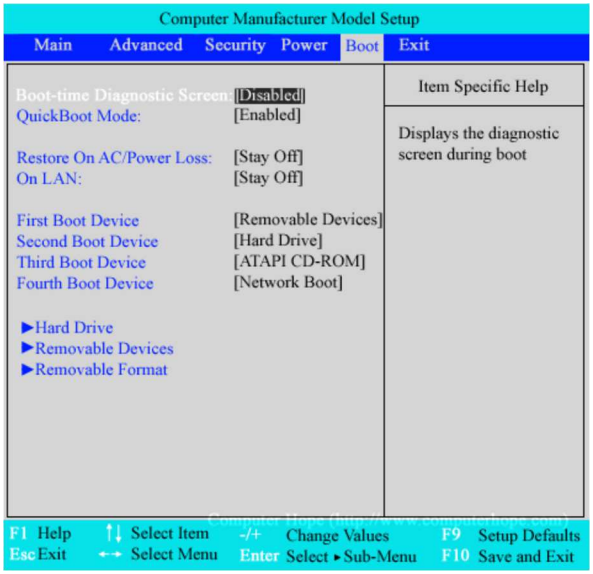
14.2. Elección y arranque del sistema operativo.

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 una unidad externa 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. En las BIOS actuales, veremos que también podemos indicarle que arranque desde un puerto USB, desde un puerto SATA, etc.

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, la unidad externa, 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.

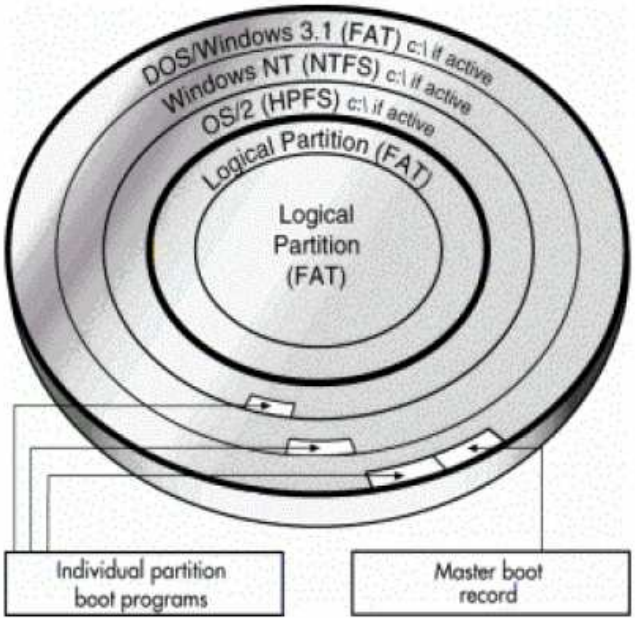
Pero con esto conseguimos indicar al sistema informático que disco duro quiero utilizar para el arranque del sistema… pero resulta que en un solo disco duro puedo tener instalado más de un sistema operativo.

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

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

14.3. Organización lógica del disco duro.

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 particiones con distintos sistemas de archivo.

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 (MBR(Master Boot Record) o GPT (tabla de particiones GUID). Esta tabla de particiones incluye una tabla donde definimos las 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.

14.4. MBR.

La tabla de particiones 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	1° entrada de la tabla de particiones	16 Bytes
+1CEh	2° entrada de la tabla de particiones	16 Bytes
+1DEh	3° entrada de la tabla de particiones	16 Bytes
+1EEh	4° 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 el 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

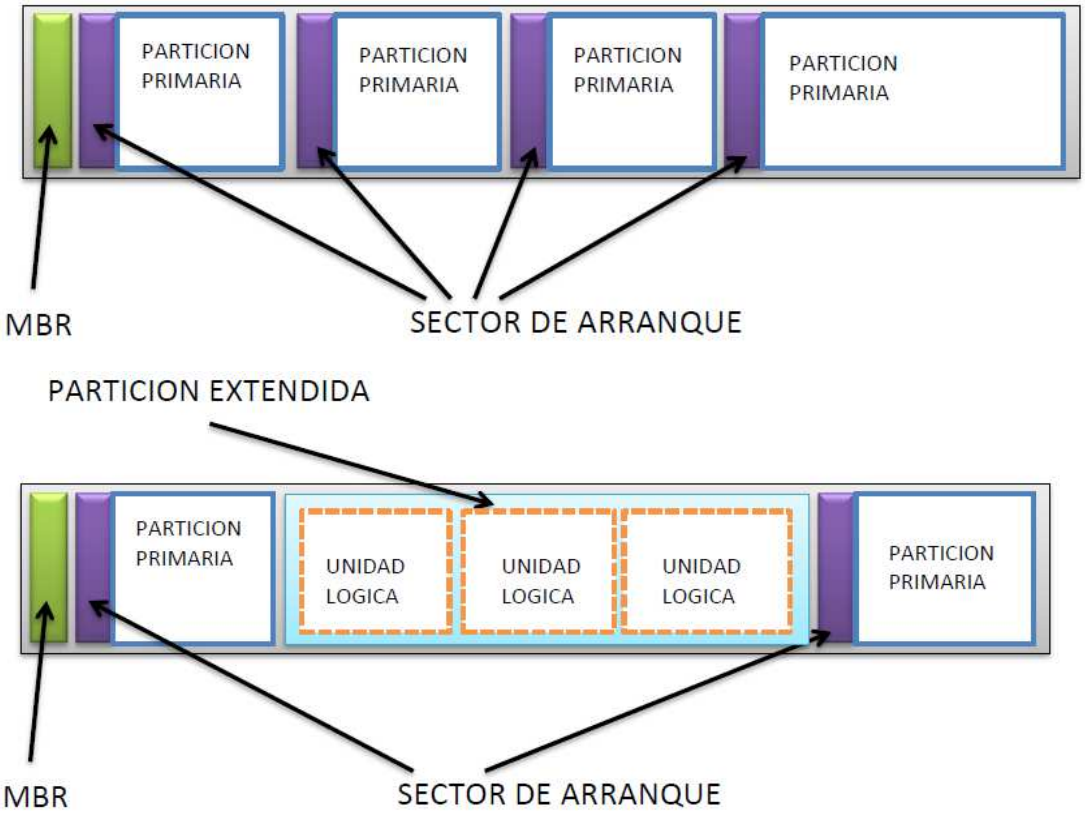
Como ya vimos anteriormente, en un disco duro puede existir un máximo de 4 particiones, sin embargo, sólo una de estas particiones puede ser extendida. (Si seguimos las recomendaciones del estándar usado para organizar lógicamente los discos duros).

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.

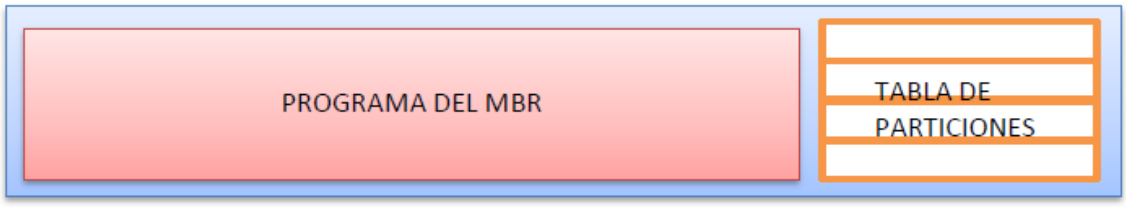
De esta manera, si dividimos un disco duro en una partición primaria (un volumen) y una partición extendida (donde creamos 10 unidades lógicas, cada una con su propio volumen) formaremos un total de 11 volúmenes (11 letras de unidad) pero solo tendremos un sector de arranque usable como tal, el de la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

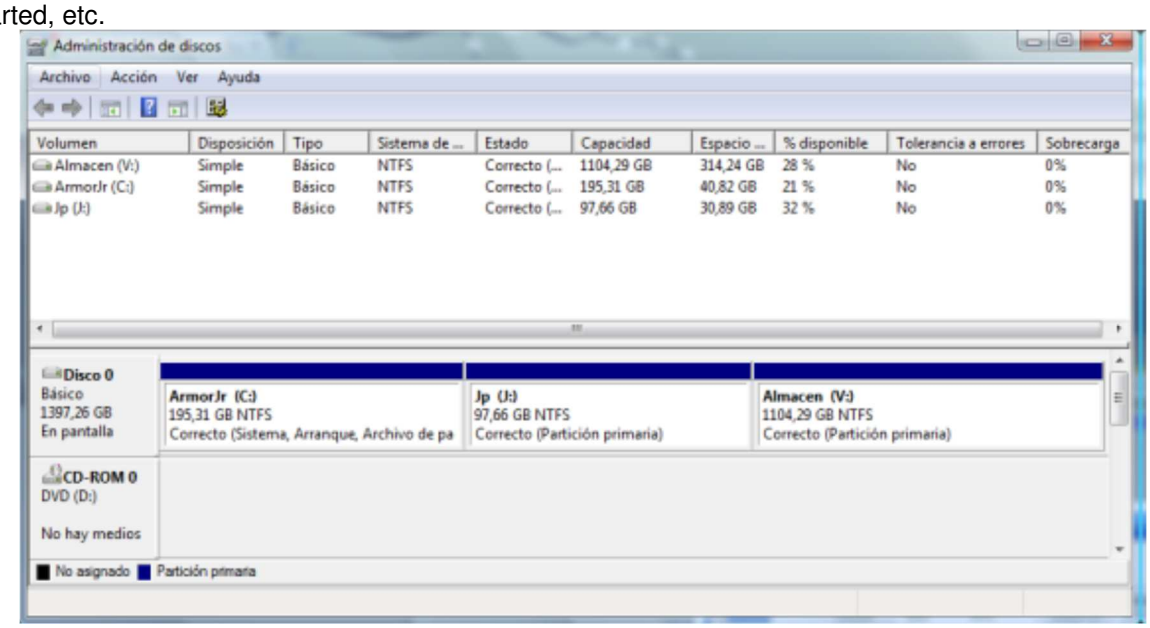


Existen diversos programas que nos permiten gestionar y retocar estos componentes del MBR.

Así, en Windows tenemos el comando FIXMBR que reinstala el programa del MBR, aunque este comando solo podemos usarlo desde la consola de recuperación.

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).

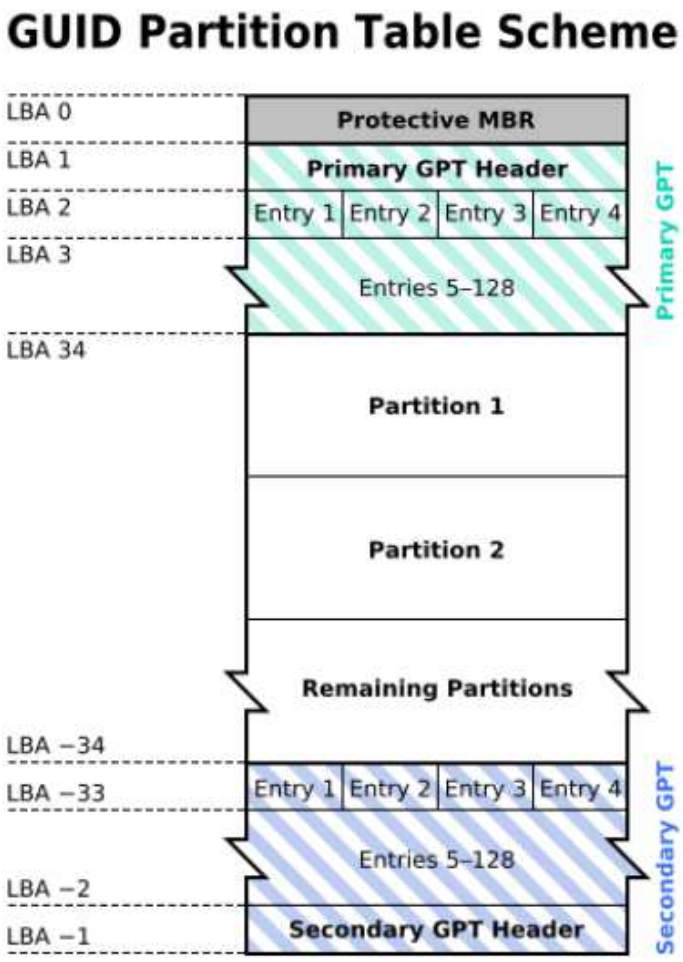
14.5. GPT.

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.



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, creerá 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 contenga 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.

14.6. Arranque de Windows 7/Vista/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 ntldr sino que se usa el Windows Boot Manager (bootmgr).

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.

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 bcedit.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 bcedit.exe y de forma mucho más fácil.

14.7. Arranque 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 frio)

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 /hybernate off
```

y volver a activarla cuando queramos con

```
powercfg /hybernate on
```

14.8. 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.

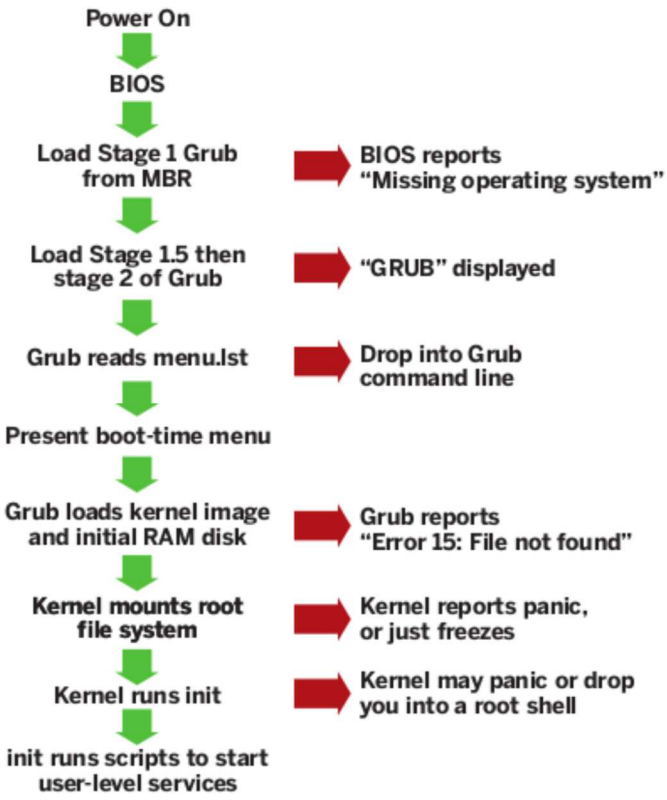
En la actualidad nos podemos encontrar con GRUB en sus versiones 1 y 2, que son algo distintas.

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 (grub) o /boot/grub/grub.cfg.
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.

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.



14.9. 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 7.

Tenemos que iniciar el sistema desde una unidad externa con el software de arranque. 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 dispositivo externo, 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 el dispositivo de externo, con el software adecuado para recuperación del grub, como puede ser por ejemplo el "súper grub disk" o "súper grub2 disk". También podemos recuperar el sistema arrancando desde el dispositivo externo, con una distribución "live". Este tema lo dejamos para cuando nos hayamos familiarizado con Linux.

14.10. 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.

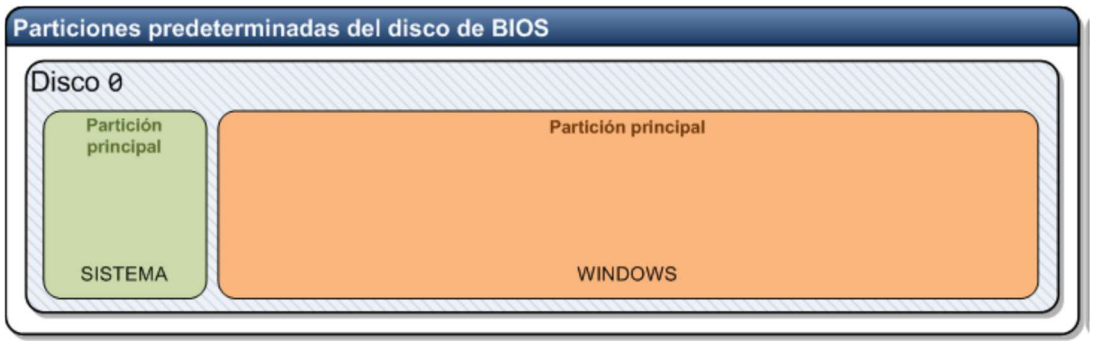
UEFI es el firmware que ahora mismo podemos encontrar en los PC comerciales, es muy extraño encontrar un equipo que no lo soporte. En BIOS de tipo UEFI únicamente podemos instalar los sistemas de 64 bits. Los de 32 bits no se pueden instalar en modo UEFI. La UEFI es un BIOS mucho más amigable que la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

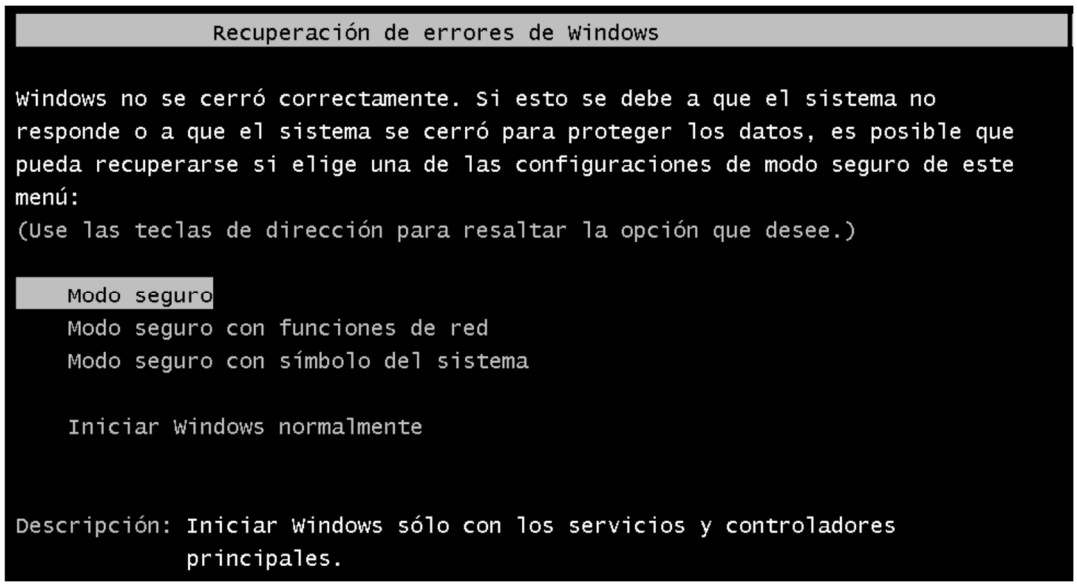


14.11. Modos de arranque a prueba de fallos.

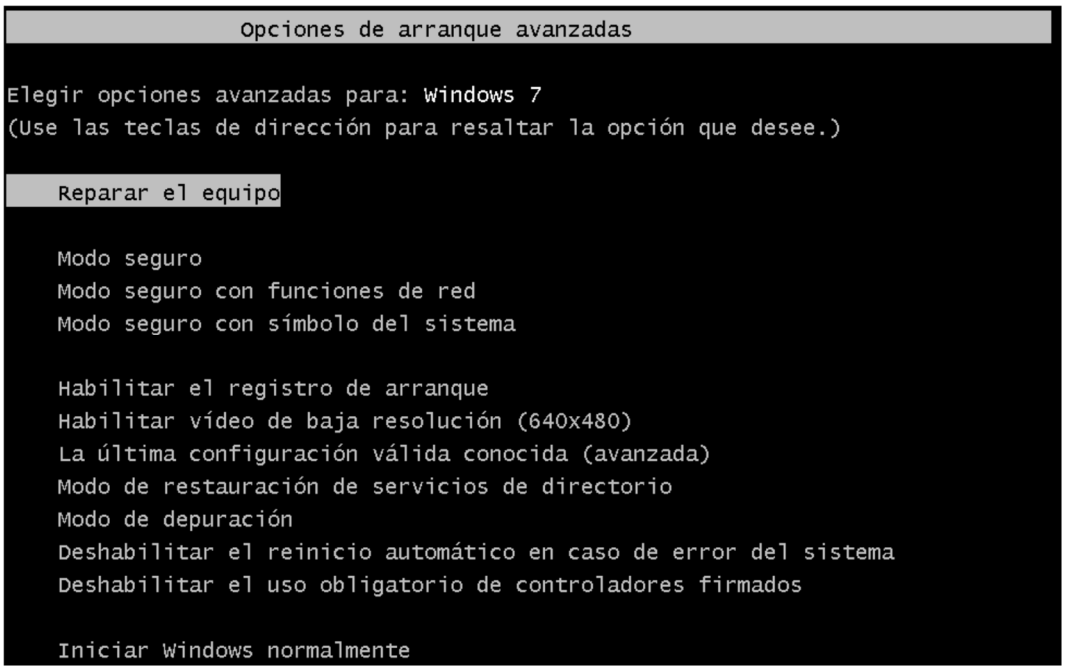
En punto anterior hemos visto cómo podemos solucionar los fallos del arranque más importantes, que conllevan sobrescribir el MBR o bien el sector de arranque. Sin embargo existen otros muchos tipos de errores que se pueden producir en el inicio del sistema operativo, y que no se pueden solucionar con 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).

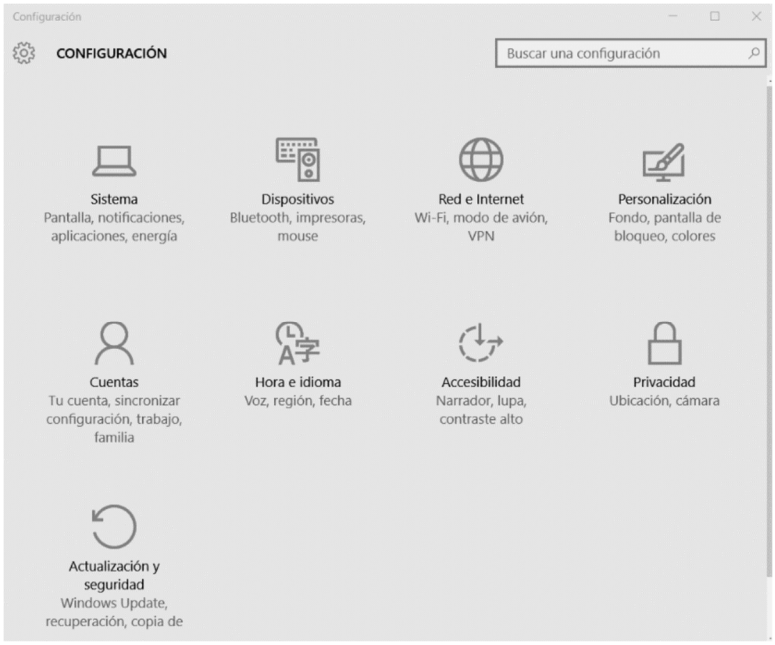


Si pulsamos F8 cuando el sistema se está iniciando, sin embargo, Windows nos presenta la siguiente pantalla:



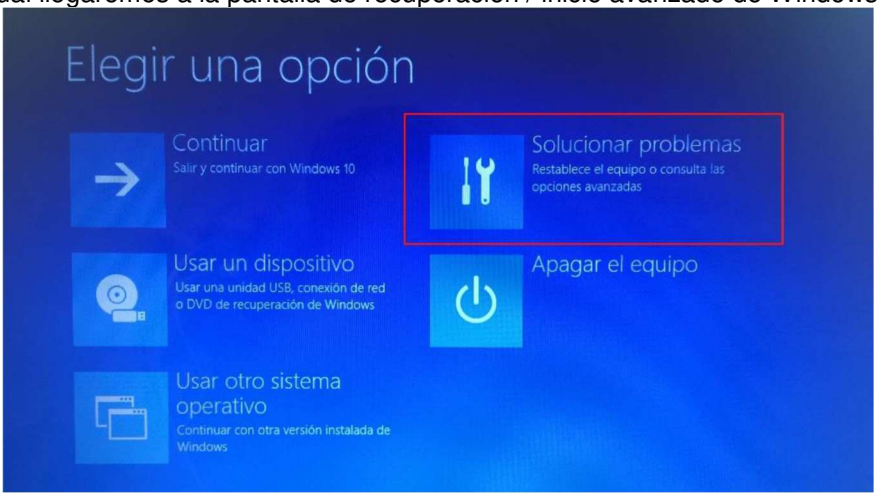
Vemos como además de los modos seguros, permite arrancar el sistema con otras configuraciones establecidas, como pueden ser con gráficos de baja resolución. Este menú aparece de una forma u otra en todas las versiones de Windows, aunque en Windows 8/10 hay que activarlo antes de poder usarlo desde el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