Sistemas Informáticos

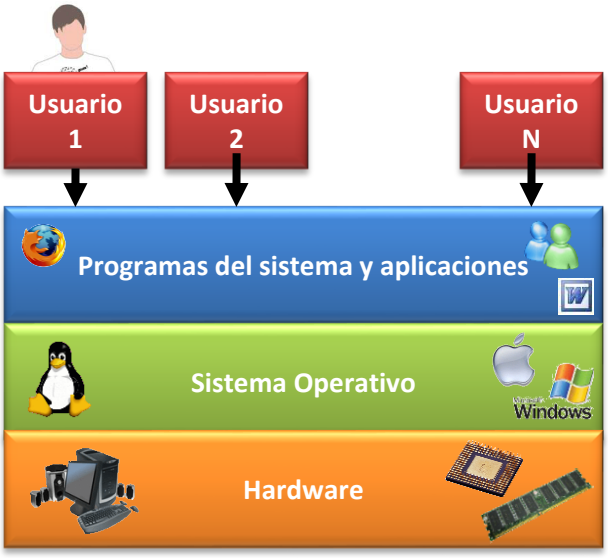
UD 5: Fundamentos de sistemas operativos.

ÍNDICE

1. El Sistema operativo 2
   1.1. Componentes 2
   1.2. Estructura de un SO 3
   1.3. Tipos 5
   1.4. Funciones de un SO 6
   1.5. Evolución 7
   1.5.1. Evolución de los SO Windows 8
   1.5.2. Evolución de los SO de Apple 9
   1.5.3. Evolución de los sistemas operativos GNU/Linux 10
   1.5.4. Estadísticas de uso de los SO en España de enero de 2015 a octubre de 2021 11
   1.6. Licencias 13
2. Gestores de arranque 14
   2.1. LILO 14
   2.2. Grub 14
   2.3. NTLDR 14
   2.4. Windows Boot Manager 14
3. Proceso de instalación de un Sistema Opeprativo 15
4. Anexos 16
   A. GESTIÓN DE PROCESOS 16
      1. Introducción 16
      2. Procesos 16
      3. Planificación de procesos 18
      4. Comunicación y sincronización entre procesos 21
      5. Procesos en Windows y en Linux 21
   B. GESTIÓN DE LA MEMORIA 22
      1. Introducción 22
      2. Problema de la reubicación 22
      3. Problema de protección de la memoria 23
      4. Problema de asignación de la memoria 23
      5. Problema de escasez de memoria. Memoria virtual 26
      6. Gestión de memoria en Linux y en Windows 27
   C. GESTIÓN DE ENTRADA/SALIDA 28
      1. Introducción 28
      2. Software de entrada/salida 29
      3. Hardware de entrada/salida 30
      4. Gestión de discos 32
   D. GESTIÓN DE FICHEROS 35
      1. Introducción 35
      2. Ficheros 35
      3. Directorios 36
      4. Métodos de asignación 37
      5. Ejemplos de sistemas de ficheros 37
      6. Gestión de ficheros en Linux y en Windows 38
   EVOLUCIÓN HISTÓRICA DE LOS SISTEMAS OPERATIVOS 38
   BIBLIOGRAFÍA 38

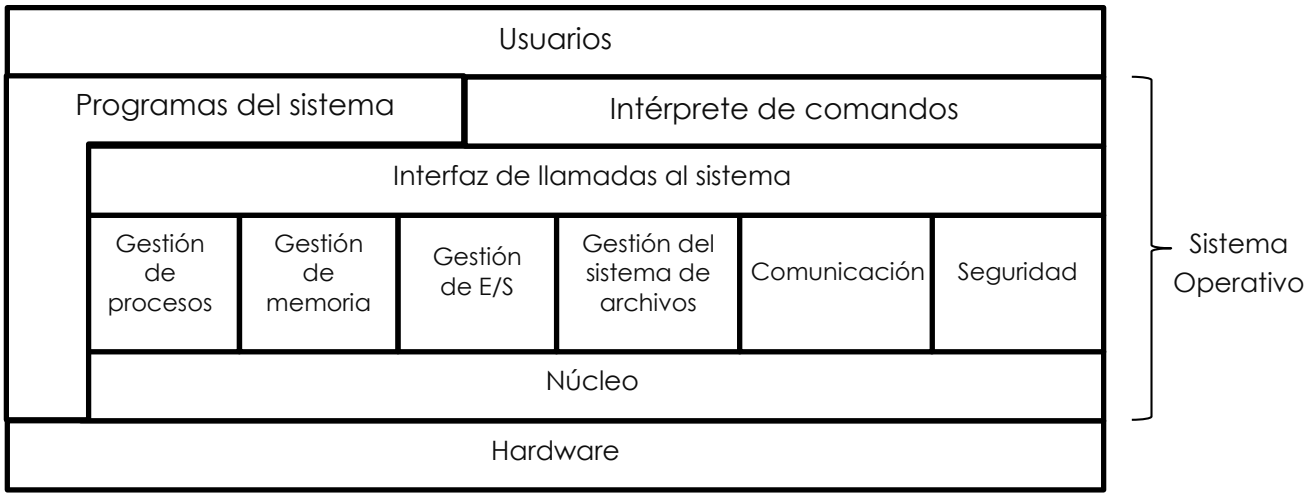
1. EL SISTEMA OPERATIVO

El **Sistema Operativo** es el **software básico** del ordenador. Es el encargado de gestionar todos los recursos hardware del SI (actúa como intermediario entre el usuario y el hardware, ocultando su complejidad con una interfaz fácil de usar y utilizándolos de forma eficiente). Una vez cargado el Sistema Operativo, ya es posible la ejecución de **aplicaciones**.



Componentes de un Sistema Informático

1.1. COMPONENTES



- Núcleo del SO: parte residente del SO, se carga al arrancar la máquina y permanece en memoria.
- Gestores de servicios: gestor de procesos, memoria, E/S, archivos y directorios, comunicación y sincronización de procesos y seguridad.
- Llamadas al sistema: interfaz formada por un conjunto de servicios que el SO ofrece a los procesos de usuario.
- Intérprete de comandos o Shell: el usuario dispone de una serie de comandos que ejecutará y el Shell interpretará proporcionando información o realizando alguna operación concreta.
- Programas del sistema: utilidades del SO que se ejecutan como procesos de usuario (fuera del núcleo). Por ejemplo: programas o ventanas para manipulación de ficheros y directorios (copy, mkdir…), programas de acceso a la red…

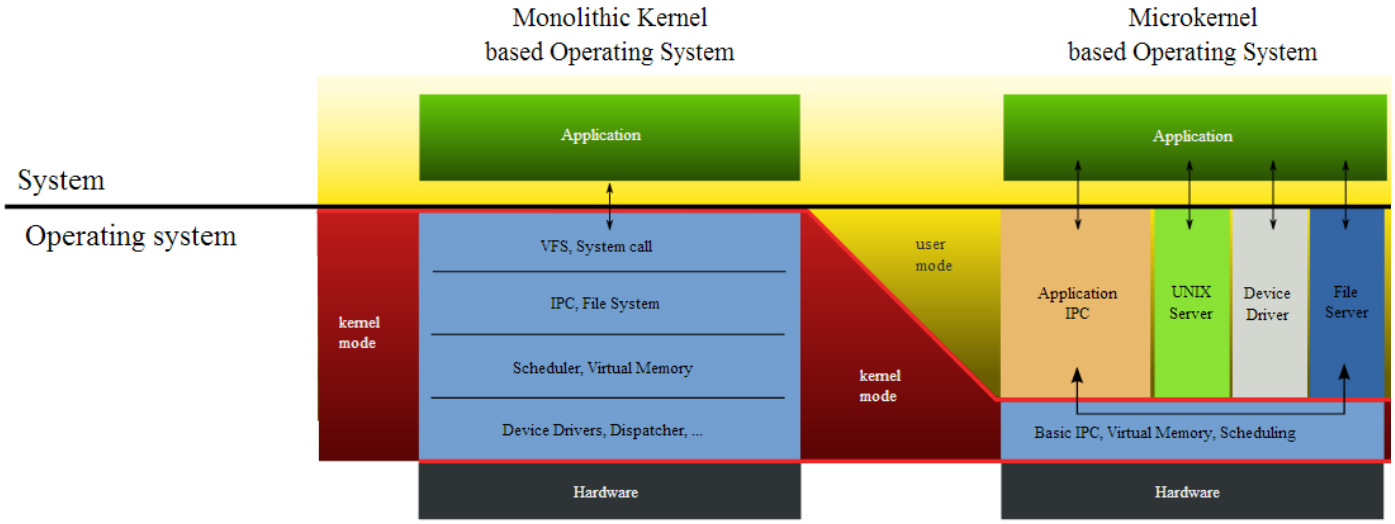
1.2. ESTRUCTURA DE UN SO

Atendiendo a su estructura interna (estrategia para organizar la arquitectura del núcleo) los SO de pueden dividir en:

- **Monolíticos**: todo el SO se ejecuta como un solo programa en modo kernel.
- **En capas o niveles**: el SO se organiza en una jerarquía de capas, cada una de las cuales tiene una determinada función. Actualmente, los SO se organizan en capas.
- **Microkernels**: el SO se divide en módulos más pequeños. Solo uno se ejecuta en modo kernel y el resto, como procesos de usuario.
- **Cliente-servidor**: los procesos se diferencian en servidores, que proporcionan ciertos servicios, y clientes, que disponen de esos servicios.
- **Máquinas virtuales**: se ejecuta un monitor de máquinas virtuales que proporciona copias virtuales del hardware al resto de procesos. En cada una de las maquinas se ejecuta un sistema operativo.
- **Exokernels**: un programa se ejecuta en modo kernel, asignando los recursos a las máquinas virtuales.
- **Híbrido**: implica que el núcleo en cuestión usa conceptos de arquitectura o mecanismos tanto del diseño monolítico como del micronúcleo

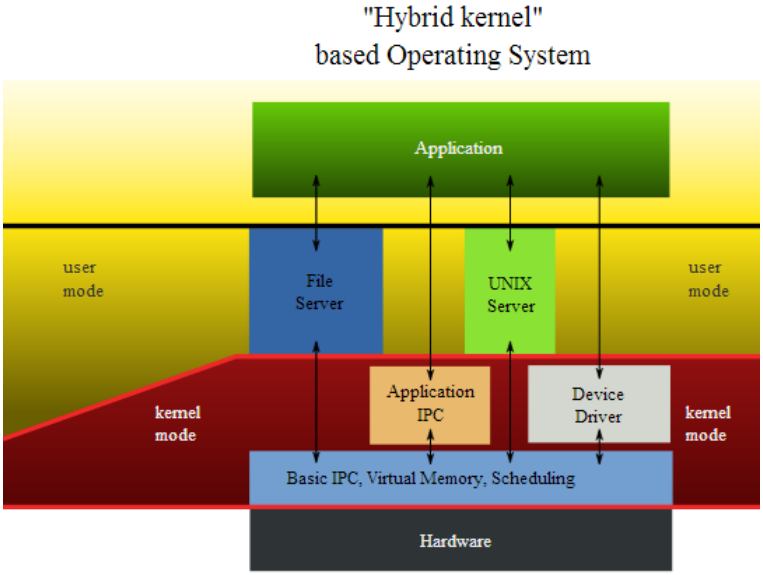
Nivel Usuario Muestra al usuario el proceso que se está ejecutando o el que se quiere ejecutar
Nivel Supervisor Se encarga de realizar la comunicación de cada proceso entre el sistema y el usuario
Nivel Ejecutivo Sobre este nivel se realiza la administración de la memoria para almacenar los procesos en páginas
Nivel Núcleo Se encarga de gestionar qué procesos llegan a la CPU para ser ejecutados

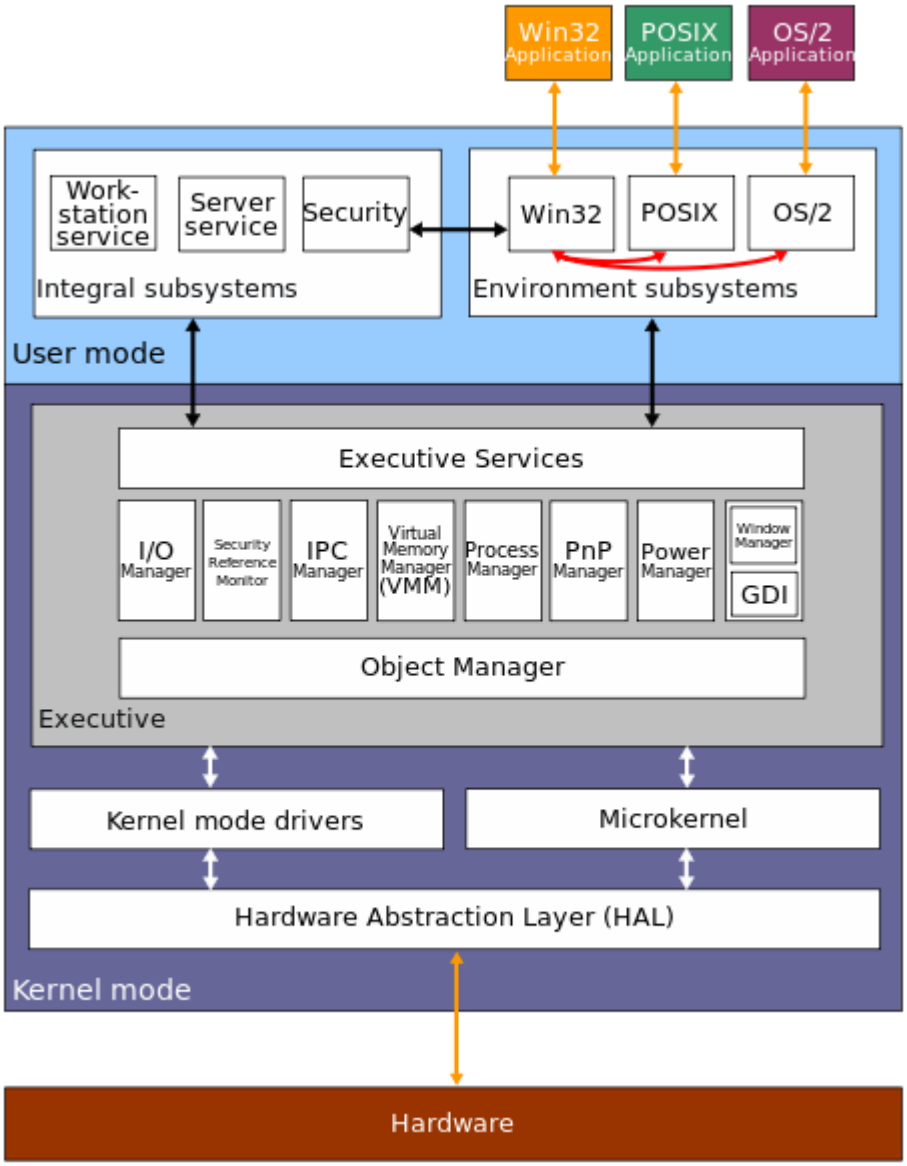
Capas/niveles de un SO



Los SO actuales, no son exclusivamente de un tipo u otro:

- **Linux**: está dividido en capas y utiliza un núcleo monolítico
- **Mac OS X**: está dividido en capas y utiliza un núcleo híbrido
- **Windows** (tecnología NT): está dividido en capas y utiliza un núcleo híbrido (en los niveles o capas superiores ejecuta ciertos procesos, servicios y las aplicaciones de los usuarios)





La arquitectura de la familia de sistemas operativos de Windows NT se basa en dos capas, (modo usuario y modo núcleo), con distintos módulos dentro de estas capas.

1.3. TIPOS

Se puede clasificar los SO en función de:

- **Nº de usuarios**:
  - **Monousuario**: solo 1 usuario puede usar los recursos del sistema simultáneamente
  - **Multiusuario**: varios usuarios pueden usar los recursos del sistema simultáneamente
    Por tanto, aunque haya más de un usuario dado de alta en el sistema, si no pueden trabajar de forma simultánea, el SO no es multiusuario.

- **Nº de procesos o tareas**:
  - **Monotarea**: solo puede ejecutar 1 tarea a la vez
  - **Multitarea** o multiprogramación: puede ejecutar varios programas a la vez

- **Nº de procesadores** :
  - **Monoproceso/monoprocesador**: el SO es capaz de gestionar solo 1 procesador, de manera que si tuviese más sería inútil. En estos SO los procesos irán alternando su ocupación en la UCP.
  - **Multiproceso/multiprocesador**: el SO es capaz de gestionar varios procesadores, de modo que puede usarlos simultáneamente para distribuir su carga de trabajo. Estos sistema trabajan de dos formas:
    - **Asimétrica**: el SO reparte las tareas, que está realizando, entre los procesadores. Determinados procesos los ejecutará siempre un procesador, y el otro procesador sólo se utilizará para realizar procesos de usuario. En este caso, es posible que un proceso esté siempre trabajando y el otro, en ocasiones, sin actividad.
    - **Simétrica**: los procesos son enviados indistintamente a cualquiera de los procesadores disponibles.

- **Tiempo de respuesta** (tiempo que tarda el usuario en obtener los resultados después de iniciar la ejecución de un programa):
  - Procesamiento **por lotes**: el tiempo de respuesta no es importante y suele ser alto. Los procesos se ejecutan secuencialmente uno tras otro. No existe interacción con el usuario. Ejemplo: copias de seguridad
  - **Tiempo compartido**: el procesador divide su tiempo entre todos los procesos (usando algoritmos de planificación como Roud Robin).Ejemplo: sistemas multiusuarios interactivos (los usuarios interactúan con el sistema)
  - **Tiempo real**: en estos SO, los procesos requieren un tiempo de respuesta muy bajo o inmediato. Ejemplos donde esto es especialmente importante: sistemas donde el tiempo de respuesta es crucial como sistemas médicos de monitorización de pacientes, sistemas bancarios, tráfico aéreo…

Sistema Operativo	Nº usuarios	Nº procesos	Nº procesadores	Tiempo de respuesta
MS-DOS	Mono	Mono	Mono	Real
Windows 9x, Me	Mono	Pseudo multitarea	Mono	Real
Windows NT Workstation, 2000 Professional, XP/Vista/7/8/10/11	Mono	Multi	Multi	Real
UNIX, Linux, Windows NT Server, Windows 2000/2003/2008/2012/2016/2019/2022 Server	Multi	Multi	Multi	Compartido

Nota: Los SO pseudo multitarea son SO capaces de cargar en memoria más de un proceso y aparentemente ejecutar más de uno al mismo tiempo. Sin embargo, estos SO no son capaces de utilizar más de 1 procesador por lo que, en realidad, solo se ejecuta un proceso en cada momento (el procesador ejecuta secuencialmente cada uno de los procesos iniciados).

1.4. FUNCIONES DE UN SO

Respecto a las funciones de todo Sistema Operativo, podemos destacar:

- Gestión de procesos
- Gestión de memoria
- Gestión de dispositivos de E/S
- Gestión del sistema de ficheros
- Gestión de la red
- Protección

A. Gestión de procesos: el SO debe ser capaz de soportar:
- La creación, suspensión, reanudación, eliminación y planificación de procesos
- La comunicación y sincronización entre procesos.

B. Gestión de memoria: el SO debe ser capaz de resolver los siguientes problemas:
- Reubicación: un proceso puede ejecutarse en diferentes secciones de la memoria física.
- Escasez: los procesos pueden necesitar más memoria de la que disponen.
- Protección: la zona de memoria asignada a un proceso debe ser privada y no debe ser 'invadida' por otros procesos
- Asignación: la memoria física debe ser repartida entre los procesos en ejecución.

C. Gestión de dispositivos de E/S: el SO debe ser capaz de manejar los diferentes periféricos existentes. Para ello, el SO debe:
- Enviar órdenes a los dispositivos
- Determinar el dispositivo que necesita la atención del procesador
- Detectar las interrupciones
- Controlar los errores
- Proporcionar una interfaz entre los dispositivos y el resto del sistema, la cual debe ser sencilla y fácil de usar y la misma para todos los dispositivos.

D. Gestión del sistema de ficheros: el SO debe ser capaz de proporcionar un modelo de trabajo sencillo con discos. Asimismo, debe ser capaz de dar solución a los siguientes problemas:
- Organización del sistema de ficheros (directorios)
- Asignación del espacio en disco a la información (de manera no contigua)
- Gestión del espacio libre y ocupado en disco

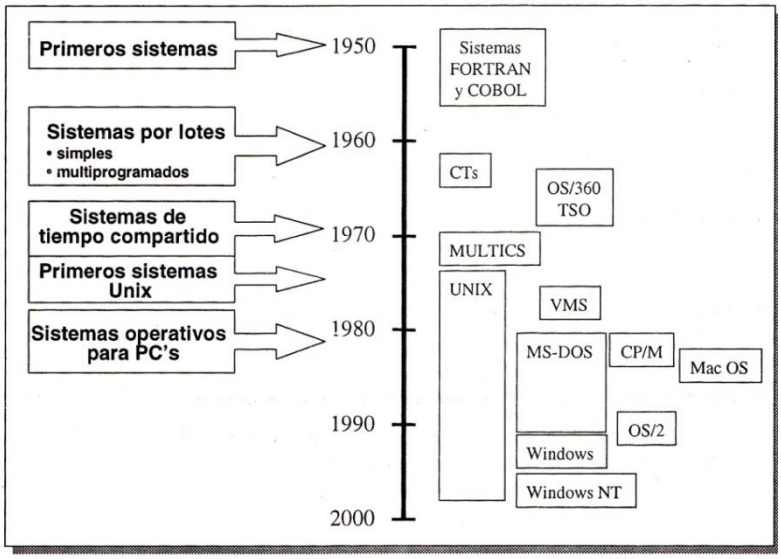
E. Gestión de la red: la cual comprende varios niveles:
- Los drivers de la tarjeta de red
- Los protocolos de comunicación (los cuales resuelven el acceso a la red y proporcionan una interfaz para comunicación entre procesos remotos)
- Las aplicaciones para uso de la red (por ejemplo: www, ftp…, las cuales son aplicaciones construidas sobre la interfaz de comunicación que facilitan el acceso a recursos remotos).

F. Protección: requiere un mecanismo para permitir/denegar el acceso de un proceso de usuario a un recurso (ficheros, dispositivos de E/S). Para ello, es necesario:
- Diseño de un modelo y política de protección para definir qué accesos son legales/ilegales.
- Implementación de un mecanismo que vigile el cumplimiento de las reglas de protección definidas.
- Mecanismo para garantizar la privacidad de la información frente a ataques de intrusos.

1.5. EVOLUCIÓN

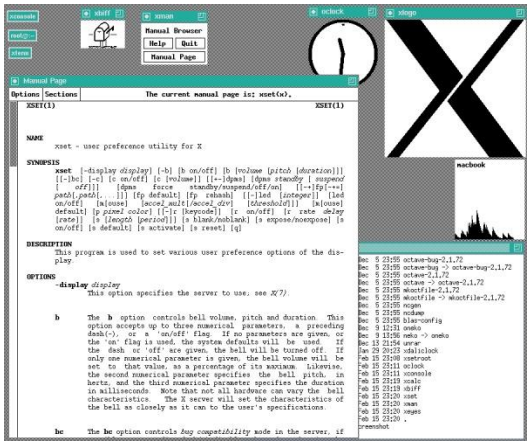
La informática tal y como se le conoce hoy día, surgió a raíz de la II Guerra Mundial, en la década de **los 40**. En esos años **no existía** siquiera el concepto de "**Sistema Operativo**" y los programadores **interactuaban directamente con el hardware** de las computadoras trabajando en lenguaje máquina (es decir, en binario, programando únicamente con 0s y 1s).

El concepto de **Sistema Operativo** surge en la década de **los 50**. El primer Sistema Operativo de la historia fue creado en 1956 para un ordenador IBM 704, y básicamente lo único que hacía era comenzar la ejecución de un programa cuando el anterior terminaba.



Evolución de los sistemas operativos

En los **años 60** se produce una revolución en el campo de los Sistemas Operativos. Aparecen conceptos como sistema **multitarea**, sistema **multiusuario**, sistema **multiprocesador** y sistema en **tiempo real.**

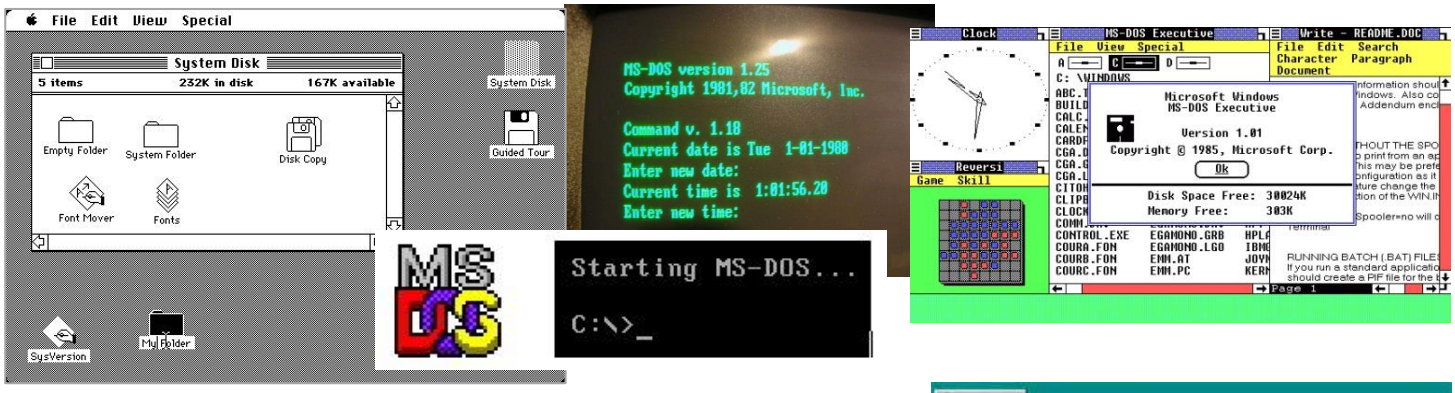


Es a finales de esta década cuando aparece **UNIX**, la **base** de la gran mayoría de los Sistemas Operativos que existen hoy en día.

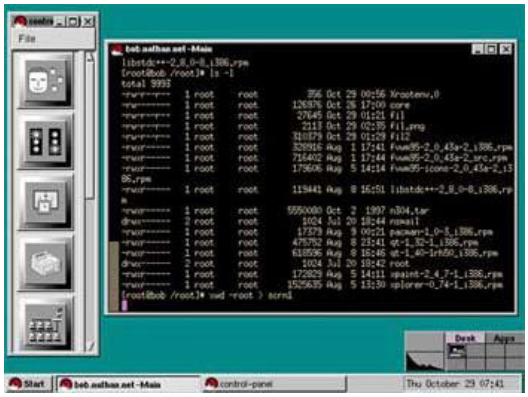
En los **años 70** se produce un **boom** en cuestión de **ordenadores personales**, acercando estos al público general de manera impensable hasta entonces. Esto hace que se multiplique el desarrollo, creándose el **lenguaje de programación C**, diseñado específicamente para **reescribir** por completo el **código de UNIX**.

Como consecuencia de este crecimiento exponencial de usuarios, la gran mayoría de ellos sin ningún conocimiento sobre lenguajes de bajo o alto nivel, hizo que en **los años 80**, la prioridad a la hora de diseñar un sistema operativo fuese la facilidad de uso, surgiendo así las primeras **interfaces de usuario**.

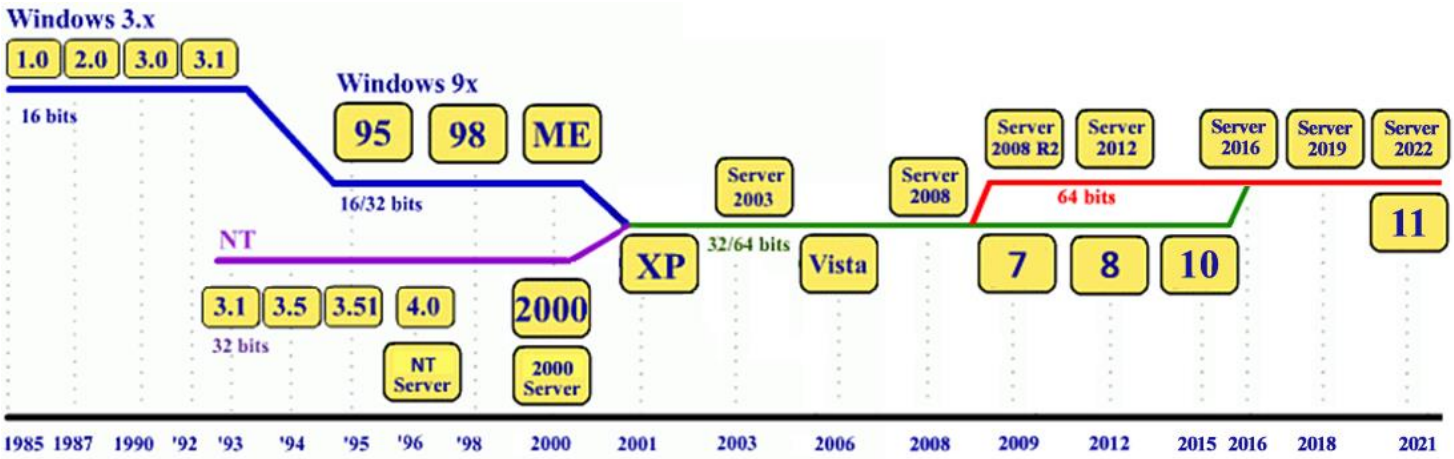
En **los 80** nacieron sistemas como **MacOS, MS-DOS, Windows**.



En la década de **los 90** hace su aparición **Linux**, publicándose la primera versión del núcleo en septiembre de 1991, que posteriormente se uniría al **proyecto GNU**, un sistema operativo completamente libre, similar a UNIX, al que le faltaba para funcionar un núcleo funcional. Hoy en día la mayoría de la gente conoce por Linux al Sistema Operativo que realmente se llama **GNU/Linux**.



1.5.1. EVOLUCIÓN DE LOS SO WINDOWS



- Tipo de núcleo: Monolítico (versiones basadas en MS-DOS). Híbrido (versiones basadas en Windows NT)
- Licencia: Microsoft CLUF/EULA (contrato de Licencia para Usuario Final): no permite que el software sea modificado, desensamblado, copiado o distribuido.
- Código cerrado.
- Última versión: Windows 11, Windows Server 2022
- Windows XP, Vista, 7, 8, 10: multitarea y monousuario
- Windows 2000/2003/2008/2012/2016/2019/2022 Server: multitarea y multiusuario
- Uso en España de Windows en octubre de 2021: 78,88%

El SO Windows de Microsoft es un software propietario, a diferencia de Linux, que es software libre, lo que significa que para instalarlo y utilizarlo en un equipo informático es necesario comprar una licencia.

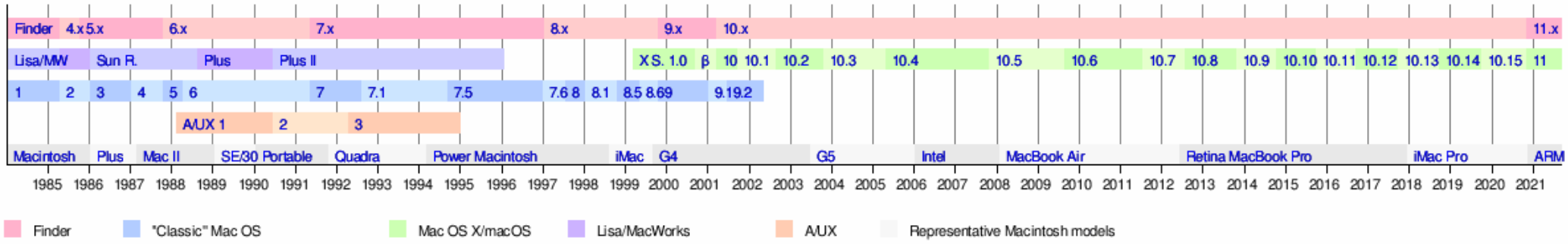
Tras el SO Mac OS de Apple, Windows fue uno de los primeros en incorporar una interfaz gráfica de usuario basada en ventanas de ahí su nombre.

En sus inicios, fue una aplicación que se ejecutaba sobre el SO de interfaz en modo texto MS-DOS, desarrollado también por Microsoft, pero posteriormente se convirtió en un auténtico SO (con Windows 95).

A lo largo de su historia, han existido varias versiones de Windows que le han ido añadiendo mejoras en su entorno gráfico, en su rendimiento, en el aprovechamiento de las características hardware del equipo y en las aplicaciones de utilidades para facilitar la labor al usuario.

En la actualidad, las versiones más utilizadas son Windows 10 y Windows 7.

1.5.2. EVOLUCIÓN DE LOS SO DE APPLE



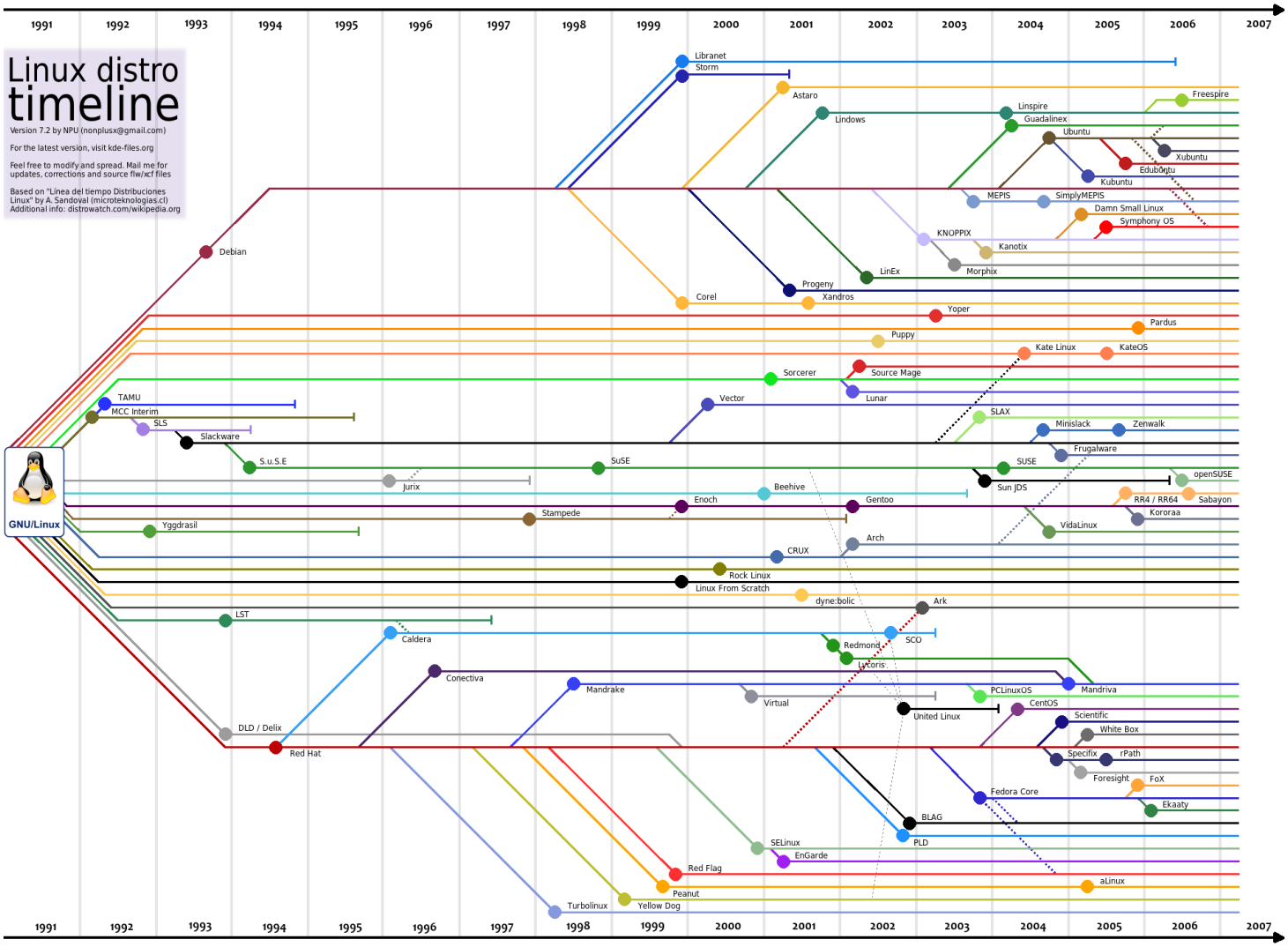
Puedes ver más detalles de cada versión de Mac OS en https://en.wikipedia.org/wiki/MacOS_version_history

Mac OS

- Tipo de núcleo: híbrido. XNU basado en Mach y BSD de Unix
- Licencia: Propietaria / Apple CLUF
- Código cerrado con componentes en código abierto (como Darwin y WebKit)
- Última versión: Mac OS 11 (Big Sur)
- Multitarea y multiusuario
- Uso en España en octubre de 2021: 13,94%

Es un sistema operativo desarrollado y comercializado por Apple que ha sido incluido en su gama de computadoras Macintosh desde 2002. Está basado en BSD. Desde la versión Mac OS X 10.5 Leopard para procesadores Intel, el sistema tiene la certificación UNIX (esto permite certificar que puede hacer funcionar ciertos software, como por ejemplo Oracle).

1.5.3. EVOLUCIÓN DE LOS SISTEMAS OPERATIVOS GNU/LINUX



Puedes ver la lista actualizada en 2021 en
https://en.wikipedia.org/wiki/File:Linux_Distribution_Timeline_27_02_21.svg

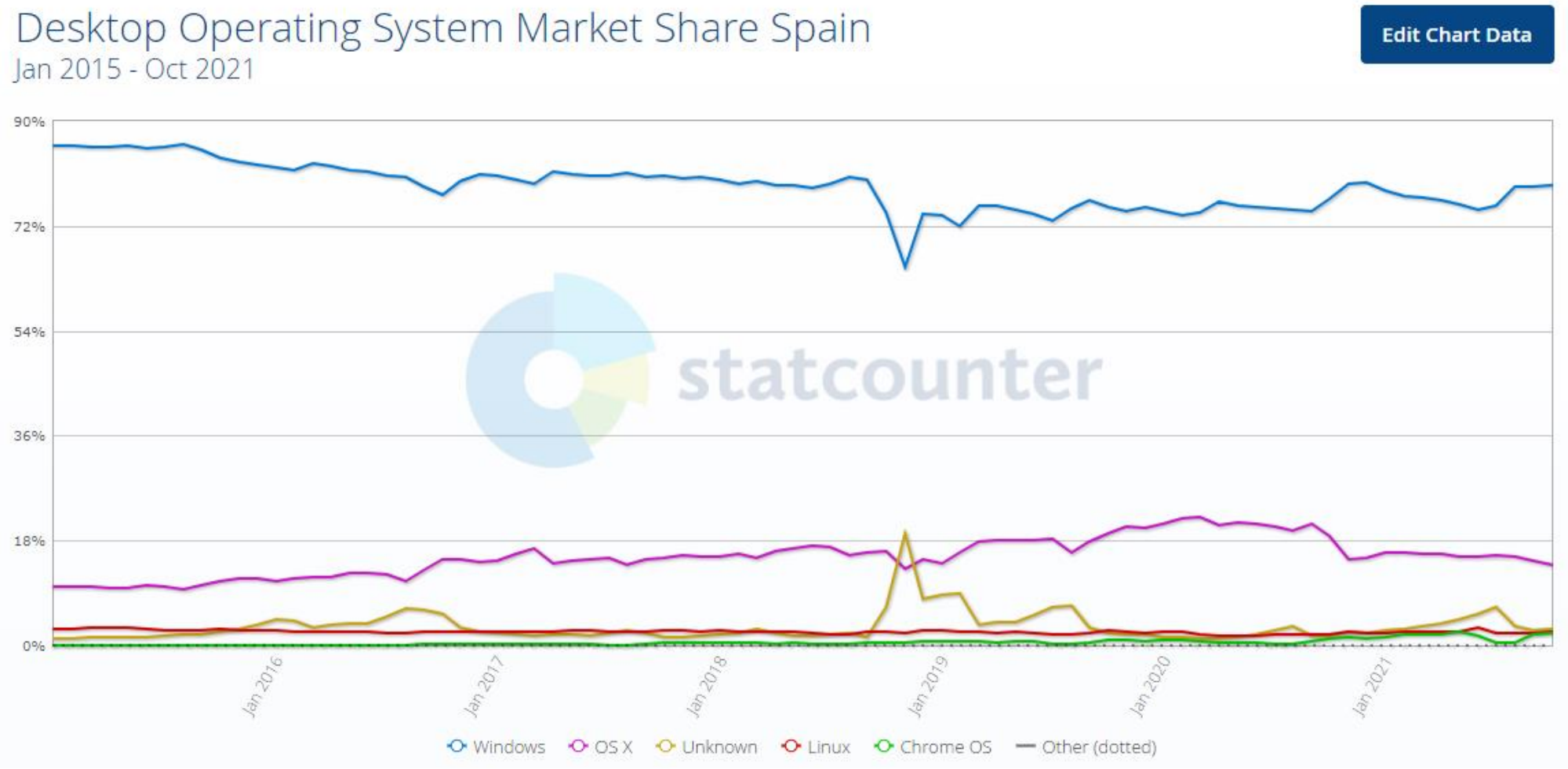
- Tipo de núcleo: Monolítico. Se basa en el SO Unix. Última versión estable Kernel: 5.14.11 (9 oct 2021).
- Software libre y código abierto
- Distribuciones de Linux actuales: Mint, Ubuntu, Fedora, Debian, Mageia, OpenSUSE
- Multiusuario y multitarea.
- Uso en España en octubre de 2021: 2,24%

Es un SO de libre distribución, por lo que se puede instalar y actualizar de forma gratuita. Además, también es de código abierto, por lo que podemos utilizar su código fuente para introducir modificaciones o mejoras.

Con el paso de los años ha ido introduciendo notables mejoras en los interfaces gráficos de usuario. Además, cuenta con muchas distribuciones y gestores de ventanas para el entorno gráfico, por lo que el usuario puede elegir la distribución que más le convenga para sus necesidades o la que más le guste, donde se puede elegir qué gestor de ventanas se quiere utilizar.

Asimismo, Linux ha mejorado bastante en cuanto al reconcomiendo del hardware sobre el que se instala y en cuanto a mejoras y novedades en el software que se puede instalar.

1.5.4. ESTADÍSTICAS DE USO DE LOS SO EN ESPAÑA DE ENERO DE 2015 A OCTUBRE DE 2021

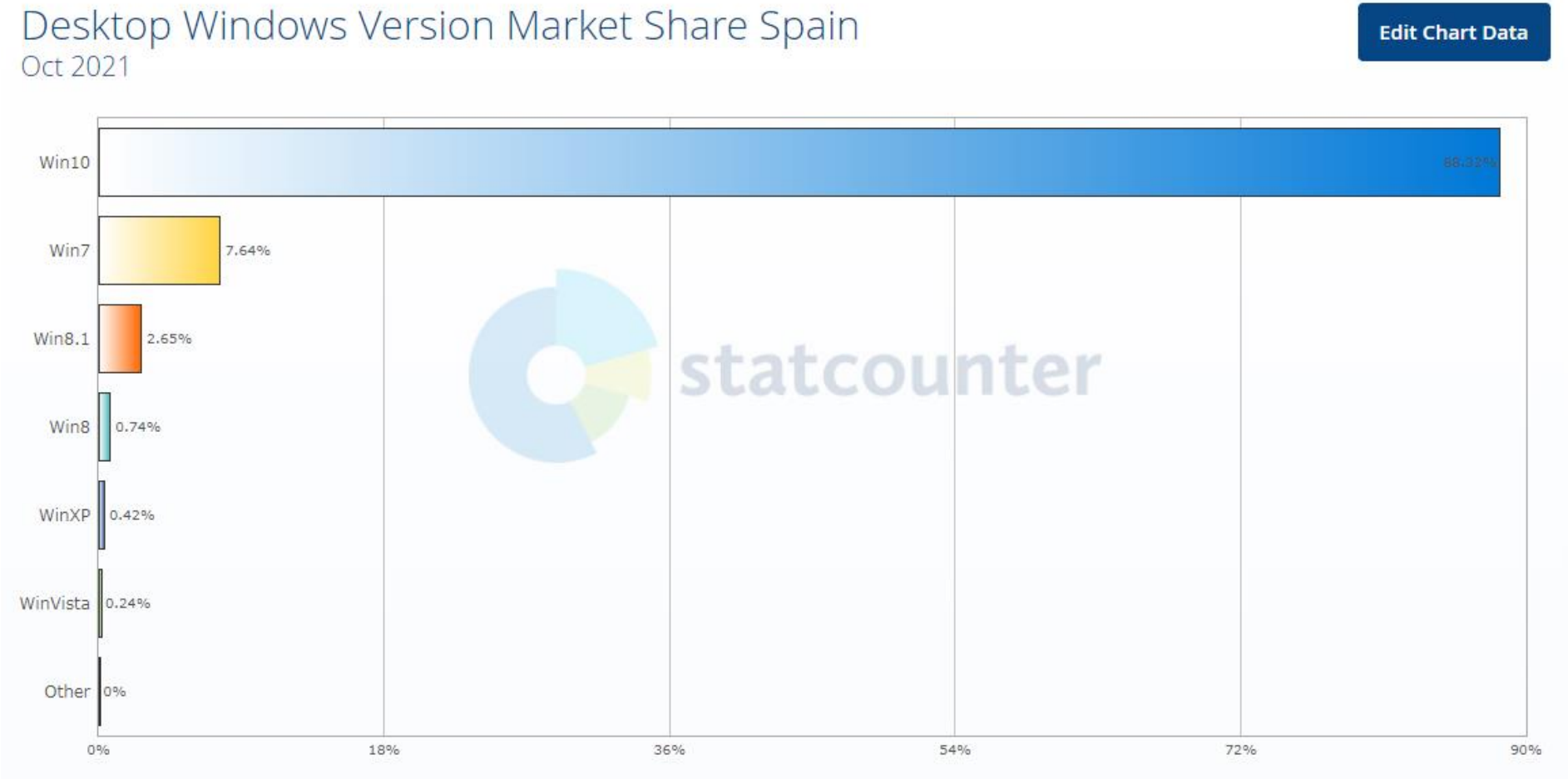


https://gs.statcounter.com/os-market-share/desktop/spain/#monthly-201501-202110

En el mes de octubre del 2021, los porcentajes de uso de los distintos SO son:

- Windows: 78,88%
- OS X: 13,94%
- Linux: 2,24%

En el caso de Windows, los porcentajes de uso en octubre de 2021 son los siguientes:



https://gs.statcounter.com/windows-version-market-share/desktop/spain/#monthly-202110-202110-bar

1.6. LICENCIAS

Según la **licencia de uso**, podemos clasificar los SO en:

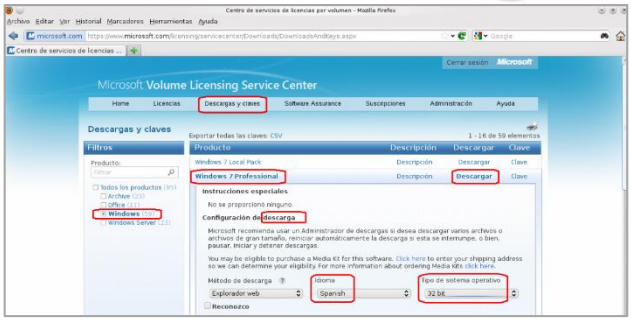
- SO **propietario**: se comercializa bajo licencia y pertenece al desarrollador, el cual posee derechos de autor y de propiedad intelectual, por lo que no puede ser copiado, distribuido ni usado sin permiso.

  En este caso, distinguimos los siguientes tipos de licencias de distribución:

  - **OEM**: Licencia unida a la venta de un equipo informático.

  - **Retail**: Licencia que compramos en cualquier tienda de SW y que podemos vender libremente cuando queramos.

  - **Por volumen o GVLK (Generic Volume License Key)**: Licencias vendidas a empresas o colegios en gran número y todas ellas con un mismo número de serie.

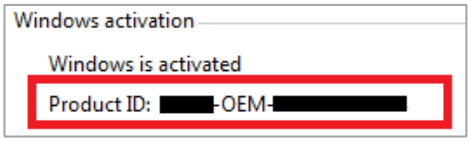


- SO **libre**: en general suele estar disponible sin coste y puede ser copiado, modificado, distribuido y usado con total libertad.

  Dentro del término libre encontramos distintos tipos:

  - **Gratuito (freeware)** o **de pago**: se puede distribuir gratuitamente o, por ejemplo, a precio de coste.
  - **Sin licencia** o **bajo licencia:** como por ejemplo, la licencia **GPL** (se puede distribuir, modificar y copiar gratuitamente, pero evitando que nadie pueda apropiarse del código).
  - **Abierto (open source):** incluye el código fuente para poderse modificar o software que no lo incluye y, por tanto, no se puede modificar.

En general, si este ID termina en OEM, entonces la licencia será del tipo OEM. De lo contrario, será del tipo RETAIL.



Con el comando "winver" también podemos ver si la licencia está vinculada a una persona física (el usuario de Windows), por lo que sería RETAIL o, de lo contrario, a un fabricante de hardware, siendo así del tipo OEM.

Normativa legal

- Los **derechos de autor** están protegidos en España por la Ley de Propiedad Intelectual, aprobada por el Real Decreto 1/1996, de 12 de abril, cuyo Libro Primero, Título VII, lleva por título "Programas de ordenador".
- Los **datos personales** y **derechos digitales** están protegidos por la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales (LOPD-GDD).

2. GESTORES DE ARRANQUE

2.1. LILO

Lilo ("Linux Loader") es un gestor de arranque que permite elegir, entre sistemas operativos Linux y otras plataformas. En las primeras distribuciones de Linux, LILO era el gestor de facto utilizado para arrancar el sistema. En la actualidad es una segunda opción en favor del gestor de arranque GRUB.

2.2. GRUB

GNU GRUB es el software que la mayoría de distribuciones GNU/Linux utilizan como gestor de arranque. Como gestor de arranque, es el primer programa que se carga del disco duro en el proceso de arranque, por eso se suele instalar en el sector de arranque del disco duro. En el directorio /boot se encuentran los archivos que el gestor de arranque necesita para arrancar el sistema, incluyendo el kernel de Linux.

Este gestor de arranque muestra al usuario un menú con todos los SO Linux y Windows que detecte en el equipo (cosa que no ocurre con el gestor de arranque de Windows que solo detecta a los SO de Microsoft). Por tanto, GRUB nos permite tener instalados varios SO y varias versiones de ellos y al arrancar el ordenador nos permite elegir cual queremos arrancar. También nos permite decidir cuál queremos tener como predeterminado.

La primera versión de GRUB usaba el fichero /boot/grub/menú.lst para configurar las opciones de arranque del sistema. En cambio, GRUB 2 utiliza el fichero /boot/grub/grub.cfg, el cual se genera a partir de:

- El fichero /etc/default/grub que modifica el menú que GRUB2 presenta por pantalla. Siempre que modifiquemos el fichero /etc/default/grub es necesario actualizar el fichero grub.cfg con el comando: sudo update-grub
- Los archivos del directorio /etc/grub.d/ que determinan el orden de aparición de las entradas en el menú. Aquellos con número menor se ejecutan antes.
- Los archivos grub-install, grub-setup, grub-mkconfig… del directorio /usr/sbin/

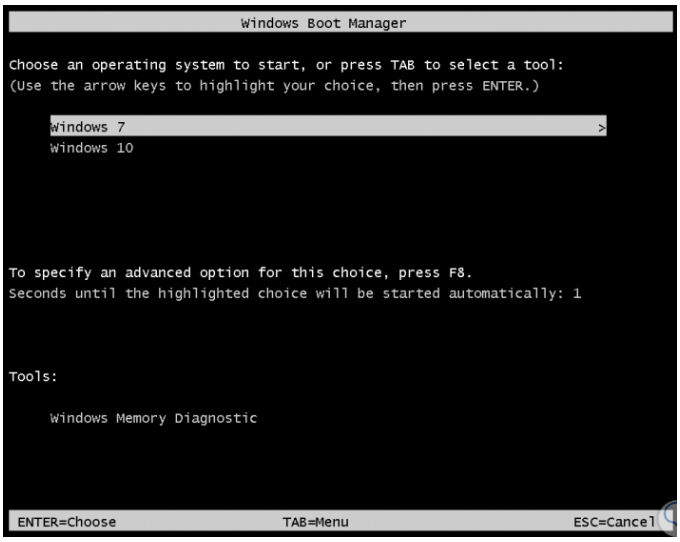
2.3. NTLDR

NTLDR (NT Loader) es el archivo encargado del arranque del Sistema Operativo en las primeras versiones de Microsoft Windows NT, incluyendo Windows XP y Windows Server 2003 (a partir Windows Vista se usa BOOTMGR).

NTLDR requiere, como mínimo, que dos archivos se encuentren en el directorio raíz del volumen de inicio:

- NTLDR, que se encarga de cargar el sistema operativo.
- boot.ini, que contiene un menú de opciones de inicio.

2.4. WINDOWS BOOT MANAGER



En Windows Vista y posteriores, el NTLDR fue reemplazado por dos componentes llamados winload.exe y Windows Boot Manager.

3. PROCESO DE INSTALACIÓN DE UN SISTEMA OPEPRATIVO

A continuación, se comentan brevemente los pasos generales en el proceso de instalación de un sistema operativo a un cliente.

1. Analizar las necesidades del cliente.
2. Revisar los requerimientos del sistema (procesador, RAM, disco duro…).
3. Obtener el SO y comprar su licencia (en caso de ser necesario).
4. Comprobar la compatibilidad del SO con el software del cliente.
5. Hacer una copia de seguridad de los datos en caso de ser necesario.
6. Arrancar desde el disco de instalación. Es posible que sea necesario modificar el orden del boot en la BIOS.
7. Esperar a que el programa de instalación cargue.
8. Configurar opciones básicas del proceso de instalación, como el idioma o el teclado.
9. Escribir la clave del SO en caso de ser necesario.
10. Escoger el tipo de instalación.
11. Formatear particiones del disco duro.
12. Configurar opciones del sistema operativo, como información personal, la hora…
13. Esperar para completar la instalación.

4. ANEXOS

A. GESTIÓN DE PROCESOS

1. Introducción

En lo que respecta a la gestión de procesos, el SO debe ser capaz de soportar:

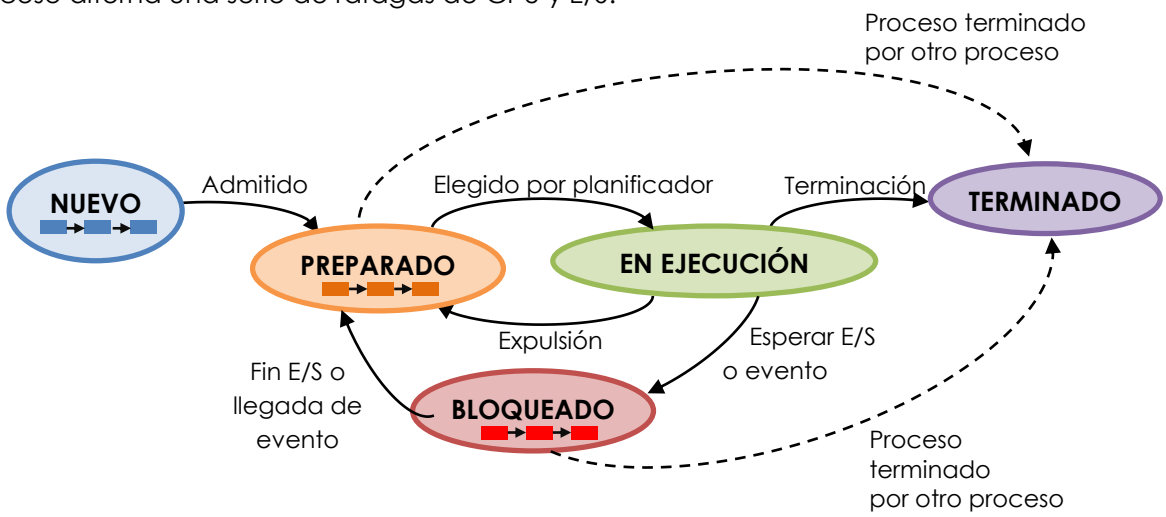
- La creación, suspensión, reanudación, eliminación y planificación de procesos
- La comunicación y sincronización entre procesos.

2. Procesos

Un **proceso** es un **programa en ejecución** que necesita estar **cargado en memoria** y disponer de **recursos** (CPU, memoria, archivos, dispositivos de E/S) para cumplir su objetivo. Se trata de una entidad activa. Mientras que los **programas** son un conjunto de **archivos** que están **almacenados** en algún **dispositivo de almacenamiento** (disco duro, pendrive…) y cuyo código fuente está escrito en algún lenguaje de programación. Cuando este conjunto de archivos se ejecutan, entonces pasan a ser un proceso.

2.1 Estados

El estado de un proceso se define por su actividad actual, cambiando a medida que se ejecuta. La ejecución de un proceso alterna una serie de ráfagas de CPU y E/S.



Estados de los procesos

Transiciones más importantes:

- Nuevo-Preparado: el SO está preparado para admitir un proceso más.
- Preparado-Ejecución: El planificador escoge un proceso para la ejecución.
- Ejecución-Preparado: El proceso en ejecución es interrumpido y expulsado del procesador porque ya ha consumido su tiempo asignado o porque otro proceso de mayor prioridad está esperando.
- Ejecución-Bloqueado: El proceso abandona voluntariamente la CPU y espera a un evento externo.
- Bloqueado-Preparado: Finaliza el evento que estaba esperando el proceso y pasa al estado preparado.
- Ejecución-Terminado: El proceso termina su ejecución (terminación normal)
- Preparado/Bloqueado-Terminado: El proceso es eliminado (terminación anormal).

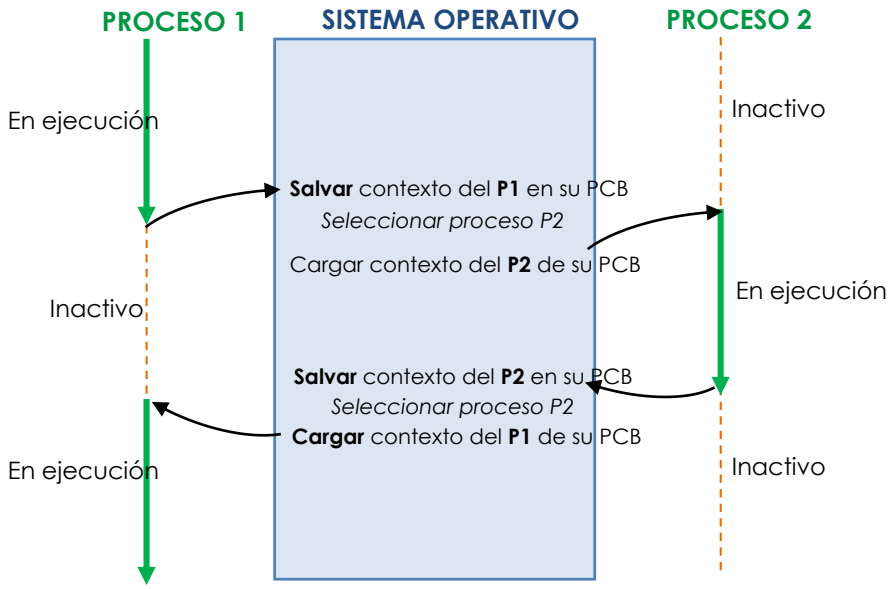
2.2 Implementación

Cuando un proceso se ejecuta, el SO le asigna un espacio de direcciones de memoria (que contiene las instrucciones, los datos y la pila (estructura para almacenar y recuperar datos del proceso) y lo añade a una **tabla de procesos**.

El S.O guarda en la tabla de procesos **por cada proceso** una estructura de datos llamada **Bloque de Control de Proceso** (PCB) que almacena la siguiente información:

- **Identificación** de proceso (del proceso en sí (PID), del proceso padre(PPID) y de usuario)
- **Información de estado** del proceso: preparado, en ejecución, bloqueado…
- **Prioridad** del proceso
- **Dirección de memoria** donde se ha cargado el proceso
- **Otros** (recursos utilizados, valores de los registros del procesador, propietario, permisos…)

Para dar sensación de ejecución simultánea o **multiprogramación**, el tiempo de CPU debe repartirse entre los procesos. Esto implica **cambios de contexto** que consisten en quitarle la CPU al proceso "en ejecución" y asignársela a otro en estado "preparado". Esta operación la realiza un componente del S.O. llamado dispatcher o **planificador a corto plazo** y en ella se guarda el contexto del proceso en ejecución en su PCB y se restaura el contexto del nuevo proceso a ejecutar mediante su PCB.



Cambios de contexto

Los cambios de contexto pueden suponer una sobrecarga si se utilizan con mucha frecuencia. En general, suelen producirse cuando un proceso finaliza, es expulsado o se suspende.

2.3 Hilos

En los actuales S.O's un proceso puede tener internamente varias actividades concurrentes llamadas hilos de ejecución o threads. Son como "miniprocesos", unidades pequeñas en las que se divide un proceso, cada una de las cuales realiza una acción.

2.4 Procesos y servicios

Los procesos que se ejecutan en un ordenador pueden estar en primer o segundo plano. Los de **primer plano** interactúan con los usuarios, mientras que los de **segundo plano** realizan una función específica sin que interactúe el usuario. Además, los procesos pueden pertenecer al usuario o ser propios del SO.

De este modo, el sistema operativo puede crear procesos, algunos de los cuales residen en **segundo plano** y se llaman **servicios en Windows** y **demonios en Linux**. El administrador del sistema puede iniciar, detener, pausar, reanudar… estos servicios. Los procesos que pertenecen al sistema se ejecutan en el modo kernel o modo privilegiado (podrán acceder a cualquier recurso).

Otros procesos son los que **crea el usuario**, donde éste **interactúa** con ellos, es decir, se ejecutan en **primer plano**, como, por ejemplo, el Firefox. Estos procesos se ejecutan en el llamado "modo usuario" (con restricciones de acceso a los recursos hardware).

3. Planificación de procesos

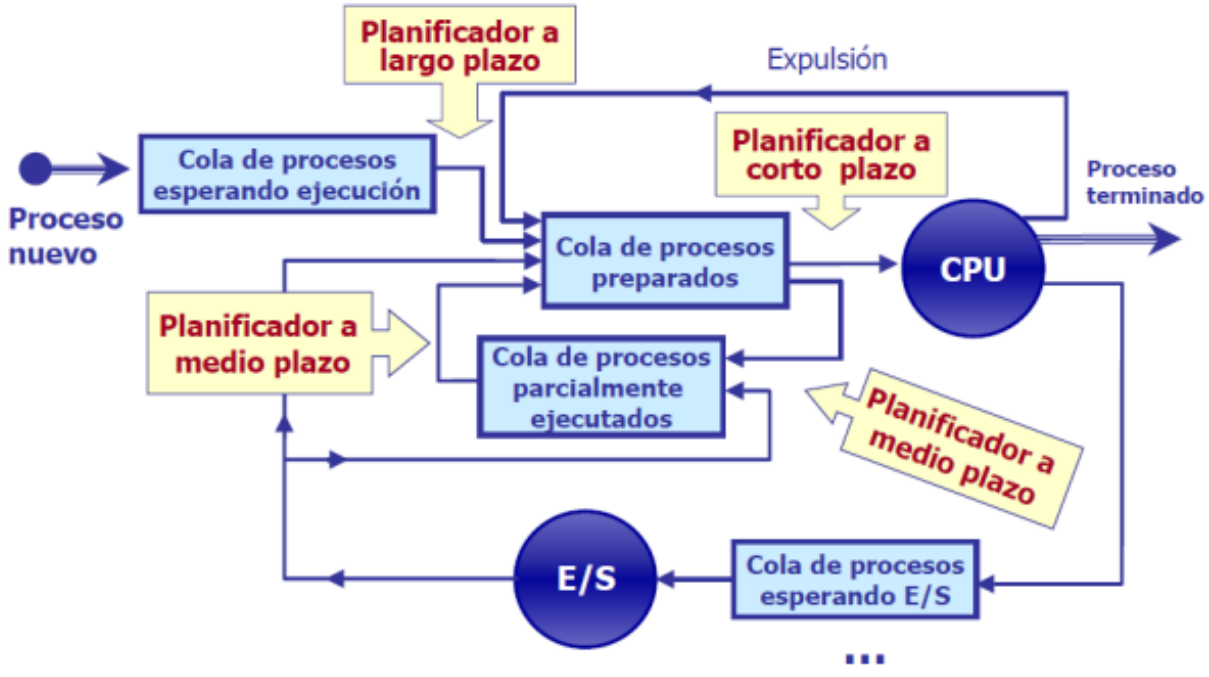
Como hemos comentado, en un S.O. multitarea la **multiprogramación** se consigue mediante la ejecución alternativa de procesos.

La parte del S.O. encargada de seleccionar el siguiente proceso para ejecutar es el **planificador** y el algoritmo que usa es el algoritmo de planificación.

3.1 planificadores y colas

Los principales planificadores de CPU son:

- Planificador a **largo plazo**:
  - Selecciona procesos de la cola de esperando ejecución y los carga a memoria.
  - Controla el grado de multiprogramación. Es importante que elija un conjunto equilibrado de procesos.
  - Se ejecuta con poca frecuencia.
- Planificador a **corto plazo**:
  - Selecciona entre los procesos preparados en memoria y les asigna la CPU.
  - Se ejecuta con mucha frecuencia.
- Planificador a **medio plazo**:
  - Decide qué proceso pasa de la memoria principal a la secundaria (memoria virtual) o viceversa.



El S.O. enlaza los PCB's de los procesos que están en el mismo estado a las diversas colas que puedan existir.

Políticas principales para la planificación:

- Planificación no apropiativa: (algoritmos no expulsivos)
  - Los procesos se ejecutan hasta que terminan o se bloquean.
  - Sencillo de implementar.
  - Rendimiento negativo en general.
- Planificación apropiativa: (algoritmos expulsivos)
  - Los procesos pueden ser expulsados de la CPU.
  - Mayor coste de implementación. Necesitan soporte hardware adicional (relojes)
  - Mejora el servicio y evita monopolización de la CPU.

3.2 Algoritmos de planificación

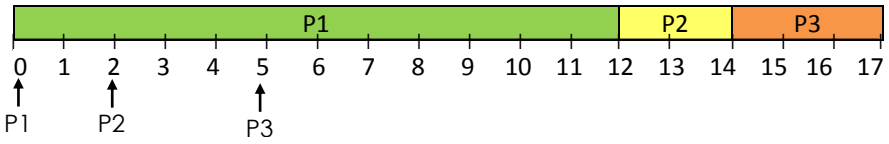
Eligen el siguiente proceso para entrar en la CPU. Los principales algoritmos de planificación son:

- **FCFS (First Come First Served)**→cola del supermercado
  La CPU es asignada a los procesos en el mismo orden que lo solicitan. Es un algoritmo no expulsivo.

  Ventajas:
  - Sencillo de implementar (cola FIFO).

  Inconvenientes:
  - Mal tiempo medio de espera
  - Efecto convoy (procesos con largas ráfagas de CPU retrasan a procesos con ráfagas cortas).
  - No válido para procesos interactivos

Proceso	Instante llegada	Tiempo CPU
P1	0	12
P2	2	2
P3	5	3



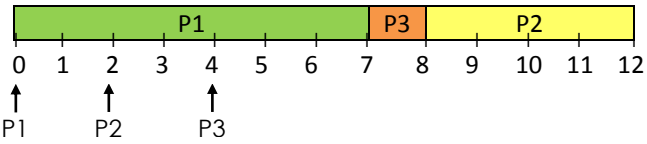
Tiempo medio de espera = (0+10+9)/3=6,3

- **SJF (Shortest Job First)**→ primero el que menos tiempo total de CPU requiere
  Se escoge el proceso de la cola de preparados con una próxima racha de CPU más corta y se ejecuta hasta que se termine o se suspenda. Si hay varios procesos con rachas de CPU iguales, se puede aplicar FIFO. Algoritmo no expulsivo.

  Ventajas:
  - Optimiza el tiempo de espera
  - Favorece los procesos orientados a E/S

  Desventajas:
  - Es costoso averiguar cuánto dura la siguiente racha de CPU
  - Inanición de los procesos con rachas de CPU largas.

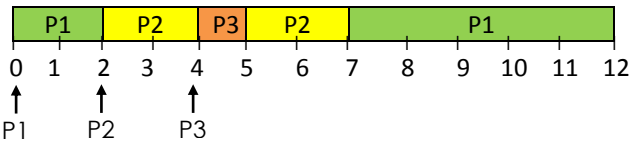
Proceso	Instante llegada	Tiempo CPU
P1	0	7
P2	2	4
P3	4	1



Tiempo medio de espera = (0+6+3)/3 = 3

- **SRTF (Shortest Remainig Time First**)→primero al que menos tiempo de CPU le queda para acabar.
  Versión apropiativa de SJF (como el SJF, solo que se puede echar a los procesos)

Proceso	Instante llegada	Tiempo CPU
P1	0	7
P2	2	4
P3	4	1



Tiempo medio de espera = (5+1+0)/3 = 2

- **<u>Planificación por prioridades</u>**→primero el que tiene más prioridad
  Cada proceso tiene asignada una prioridad. El planificador selecciona el proceso con prioridad más alta (a igual prioridad se selecciona con FCFS).

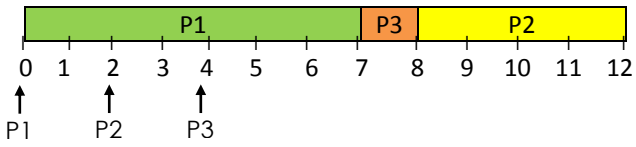
  Las prioridades pueden ser <u>dinámicas</u> (cambian con el tiempo) o <u>estáticas</u> (se mantienen).

  Inconvenientes:

  - Riesgo de inanición de procesos con prioridad baja. Una solución sería aumentar la prioridad con el incremento del tiempo de espera.

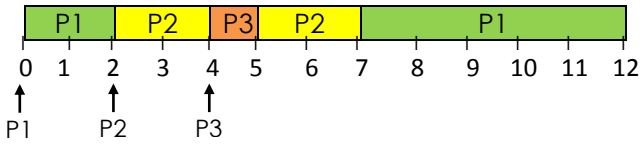
Proceso	Instante llegada	Tiempo CPU	Prioridad
P1	0	7	5
P2	2	4	10
P3	4	1	15

Prioridades sin expulsión:



Tiempo medio de espera: (0+6+3)/3=3

Prioridades con expulsión:



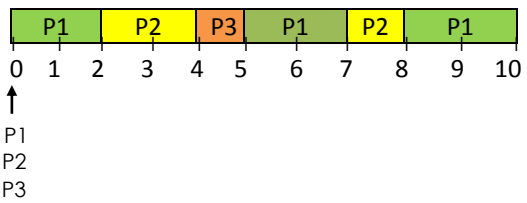
Tiempo medio de espera = (5+1+0)/3 = 2

- **<u>Planificación circular (Round Robin)</u>**→ todos el mismo tiempo por turnos
  - A cada proceso se le asigna una cantidad de tiempo de CPU llamada "<u>quantum</u>". Si el proceso tiene un intervalo de CPU mayor que el quantum es expulsado de la CPU.
  - La cola de preparados se gestiona con una política FIFO.
  - Si el valor de quantum es grande el algoritmo degenera en FCFS. Si es pequeño se generará una sobrecarga debido a cambios de contexto.
  - Es equitativo

Ej. (Quantum = 2):

Proceso	Instante llegada	Tiempo CPU
P1	0	6
P2	0	3
P3	0	1

Tiempo medio de espera= (4+5+4) / 3 = 4,3



4. Comunicación y sincronización entre procesos

Los procesos necesitan comunicarse entre sí (Ej: cat fichero | lp). El S.O. debe ofrecer mecanismos para la comunicación y sincronización de procesos.

Esta comunicación puede realizarse básicamente de dos formas:

- Por memoria común
- Por paso de mensajes

4.1 Comunicación por memoria común

Cuando los procesos se ejecutan en el mismo espacio de direcciones o tienen acceso a una zona de memoria común, es necesario sincronizar su acceso a estas zonas, ya que el resultado final depende de los instantes de ejecución de los procesos. A la parte del programa en la que se accede a una zona de datos compartida, se conoce como sección crítica.

Una solución al problema de la sección crítica sería asegurar que si un proceso utiliza una zona de memoria compartida, no deje a otros procesos utilizarla a la vez, es decir, excluir del acceso a otros, esto se conoce como exclusión mutua. El SO, por tanto, debe ordenar y controlar el acceso que hacen los procesos a la sección crítica.

4.2 Paso de mensajes

Los procesos se pueden ejecutar en espacios de direcciones independientes, por lo que no hay problema de sección crítica. La comunicación se realiza mediante el paso de mensajes de un proceso a otro, utilizando las operaciones send(dest,msg) y receive(org,msg).

Tipos de direccionamiento de mensajes

- Directa: se indica el nombre del proceso destinatario
- Indirecta: se envían los mensajes a buzones o puertos y los procesos destinatarios cogerán el mensaje de ese buzón.

Ejemplos:

- Tubos entre procesos Unix
- Modelo cliente/servidor usado en sistemas distribuidos:
    - El servidor gestiona un recurso y recibe peticiones de los clientes, las ejecuta en su nombre y les responde.
    - Los clientes envían mensajes de petición al servidor y espera respuesta

5. Procesos en Windows y en Linux

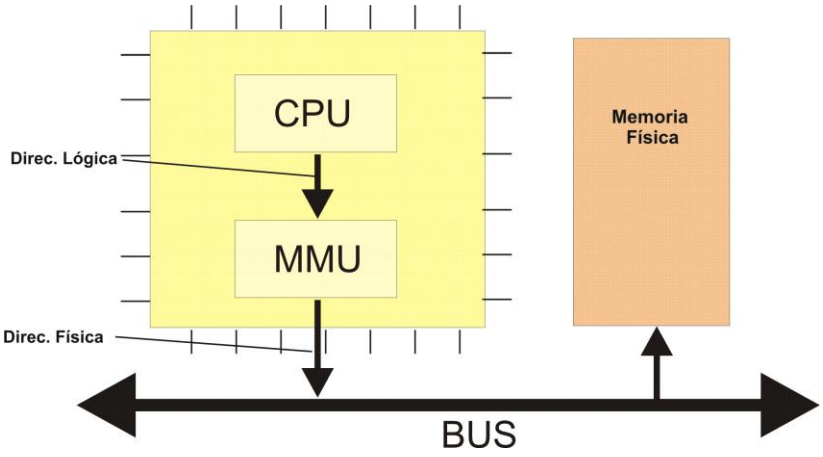
Podemos ver los procesos, así como su consumo de CPU y RAM:

- En Windows:
    - "Administrador de tareas" (apretando alt+ctrol+supr)
    - "Monitor de recursos" (ejecutar→resmon; viene por defecto en Win7)
    - "Explorador de procesos" (http://download.sysinternals.com/files/ProcessExplorer.zip)
- En Linux:
    - "Monitor del sistema" (sistema→administración→monitor del sistema)

B. GESTIÓN DE LA MEMORIA

1. Introducción

A continuación, se analizará los problemas referentes a gestión de la memoria a los que debe dar solución todo SO (reubicación, protección, asignación y escasez de la memoria), así como las técnicas utilizadas en su solución, haciendo un especial hincapié en las técnicas de asignación dispersa y memoria virtual.



En esta tarea de gestión de la memoria, existe un componente clave: la **MMU** (Unidad de Manejo de Memoria), que es un componente hardware, normalmente integrado en la CPU, responsable de manejar las peticiones de acceso a memoria que solicita la CPU.
Concretamente, la MMU es la responsable de **transformar las direcciones lógicas** o virtuales que emite la CPU a **direcciones físicas** que recibe la memoria y proteger la memoria, entre otras funciones.

Por tanto, al igual que un computador dispone de una memoria física, podemos considerar que cada proceso tiene su memoria lógica o virtual.

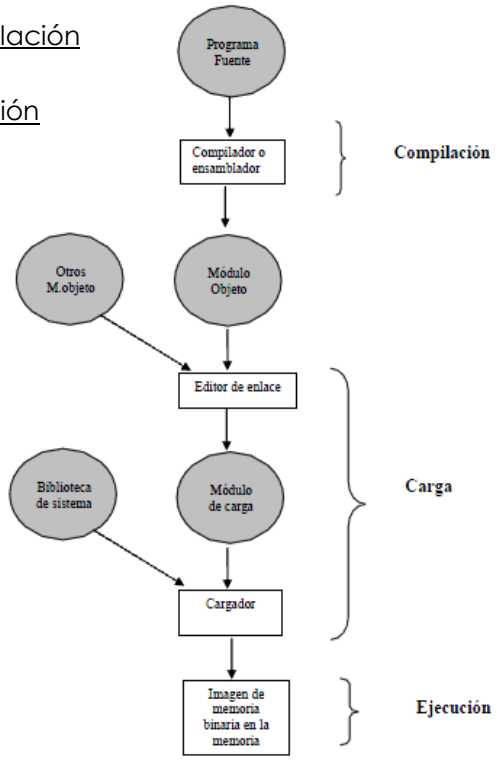
2. Problema de la reubicación

Un proceso debe poder ejecutarse en diferentes ubicaciones de la memoria física, lo cual implica que las referencias a memoria que existen en el código del programa, tienen que ser concretadas en posiciones de memoria física.

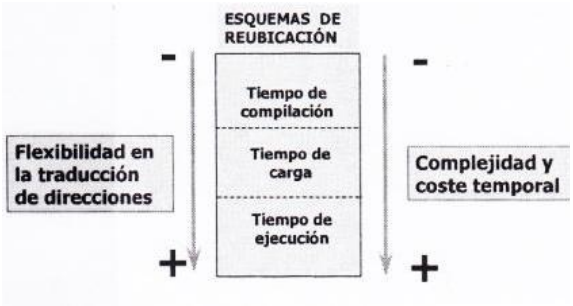
Además, si un proceso se manda de memoria principal a memoria secundaria, cuando vuelva a cargarse a memoria principal, debe poder hacerlo en otro lugar de la memoria diferente al que estaba inicialmente.

Esto puede resolverse en cualquiera de las siguientes fases:

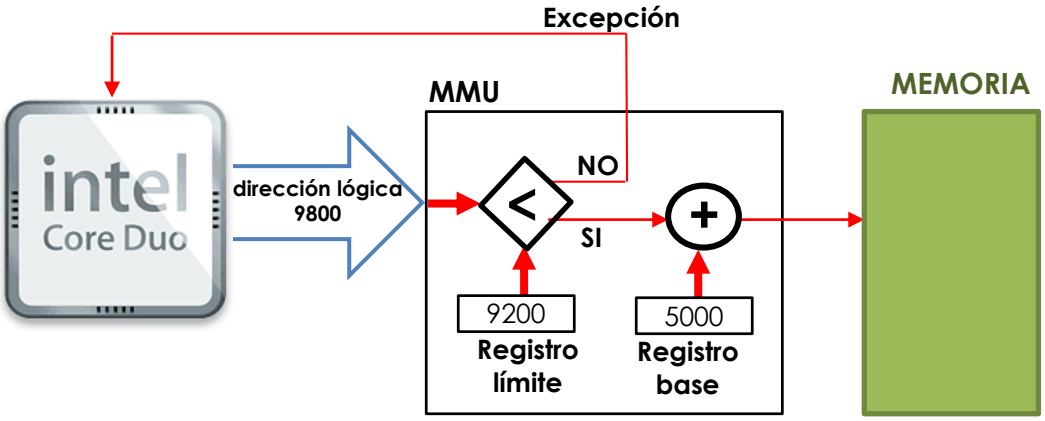
- Compilación
- Carga
- Ejecución



En el siguiente gráfico, podemos ver una comparativa de los métodos:



3. Problema de protección de la memoria



Los procesos SOLO pueden acceder a su propio espacio de direcciones físicas (salvo que el sistema permita compartir zonas de memoria). Por tanto, el SO debe evitar que un proceso invada la zona de memoria de otro proceso o del SO. Por ejemplo: usando registros base y registros límite que contienen la dirección inicial y final, respectivamente, de la zona de memoria de cada proceso.

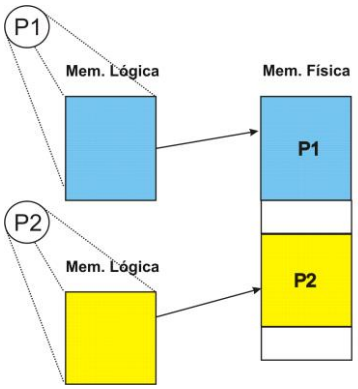
Ejemplo: supongamos que la CPU emite la dirección lógica 9800, como ésta no debe exceder de 9200 y la dirección que se ha emitido es superior, la MMU no lo permite y avisa del error a la CPU. Si la dirección lógica emitida fuera menor que el registro límite, el valor de esta dirección se sumaría al registro base, dando lugar a la dirección física de memoria a acceder.

4. Problema de asignación de la memoria

La memoria física puede ser asignada a los diversos procesos en ejecución siguiendo diversas técnicas:

- Asignación contigua
- Asignación dispersa

4.1 Asignación contigua

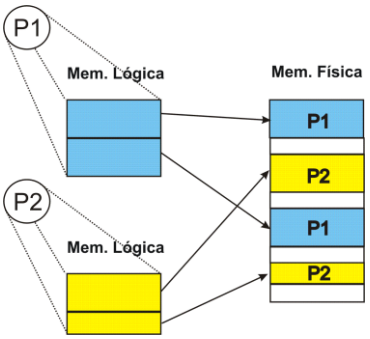


El espacio de direcciones lógicas de un proceso se mapea sobre una única zona de la memoria física: las direcciones de memoria son contiguas.

Métodos:

- Particiones fijas
- Particiones variables

4.2 Asignación dispersa



La memoria lógica se divide en fragmentos (páginas o segmentos), que se mapean sobre zonas no contiguas de la memoria física.

Técnicas de asignación dispersa:

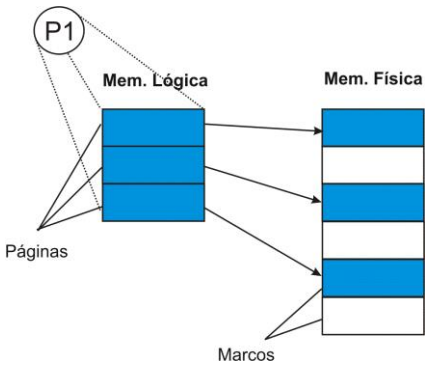
- Paginación
- Paginación multinivel
- Segmentación
- Segmentación paginada

Para implementar estas técnicas se necesita el apoyo de la MMU.

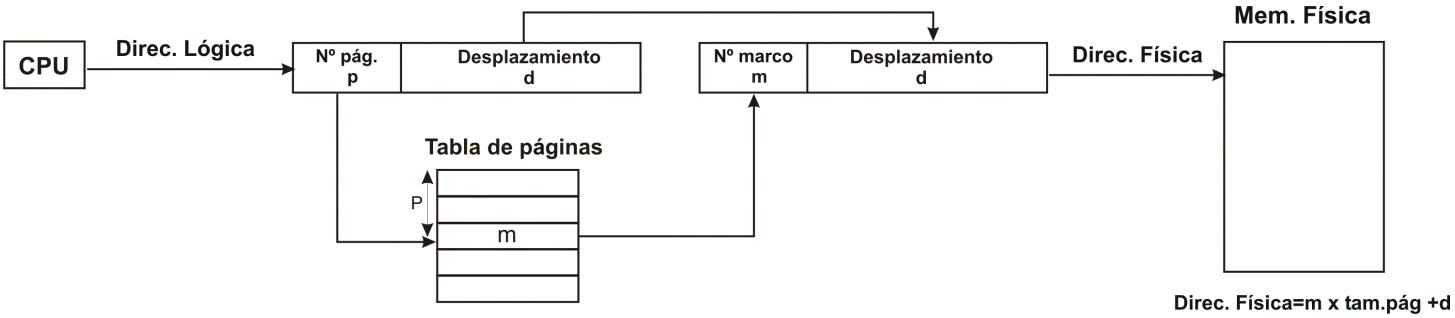
4.2.1 Paginación

- La memoria lógica de un proceso se divide en bloques de tamaño fijo denominados páginas que tienen una longitud potencia de 2 (1kb, 2kb, 4kb, 8kb).
- La memoria física se encuentra dividida en bloques de igual tamaño que las páginas denominados marcos.

- Las páginas se mapean sobre los marcos:



- A continuación, se muestra el proceso de traducción de direcciones:



- Cada proceso tiene asociada una **tabla de páginas** que contiene descriptores, uno por cada página del proceso, con la siguiente información:
  - Nº de marco físico: para traducir direcciones
  - Bit de validez: indica si la página esta en memoria física
  - Bits de protección (RWX): indica los tipos de acceso permitidos sobre el marco
  - Bit de modificado: indica si el marco difiere de su imagen en disco

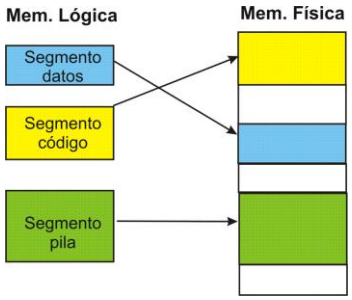
- **Análisis de la paginación:**
  - Ventajas:
    - No aparece fragmentación externa
    - Facilita la reubicación
    - Proporciona protección: no se puede acceder a páginas físicas no asociadas
    - Compartición de páginas entre procesos (ahorro de memoria).
  - Inconvenientes:
    - Posibilidad de fragmentación interna: queda un espacio no utilizado en el último marco asignado a cada proceso.
  - Respecto al tamaño de página:
    - Debe ser potencia de 2 para facilitar la traducción de direcciones.
    - Tamaños de pág. grandes: aumentan la fragmentación interna.
    - Tamaños de pág. pequeños: requieren tablas de página grandes.

4.2.2 Paginación multinivel

Esta técnica se utiliza para espacios de direcciones muy grandes, en los que la tabla de páginas puede ser excesivamente grande. Consiste en paginar la propia tabla de páginas, usando dos niveles o más de paginación.

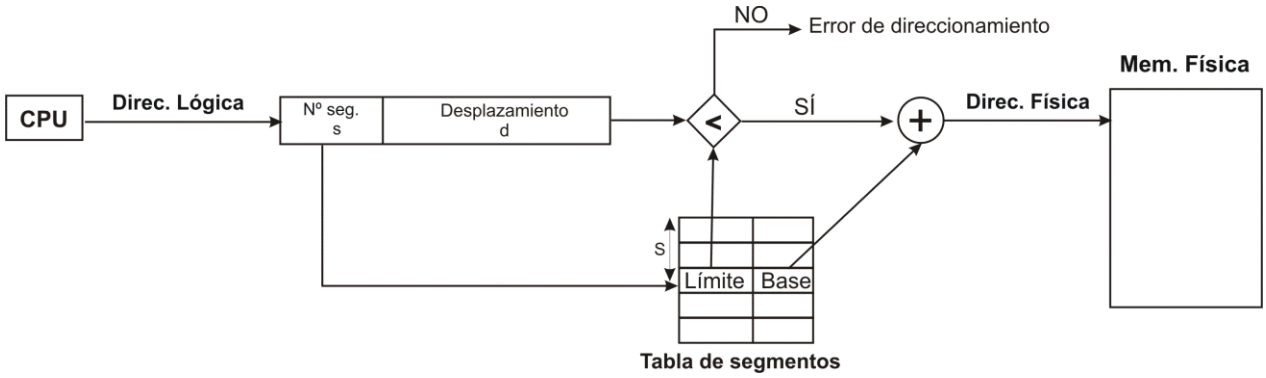
Inconveniente: Cada nivel de paginación requiere incrementar en uno el número el número de accesos a memoria para acceder a una dirección física.

4.2.3 Segmentación



- La memoria lógica de un proceso se divide en varios subespacios independientes de longitud variable denominados segmentos, por ejemplo: segmentos de código, datos o pila.
- El segmento se ubica en un área contigua de memoria física, aunque distintos segmentos pueden ir ubicados en zonas no contiguas.

- En el siguiente esquema, se muestra el **proceso de traducción de direcciones**:



- La **tabla de segmentos** tiene un descriptor por cada segmento, el cual contiene:
  - Base del segmento: dirección física del comienzo del segmento
  - Límite: longitud del segmento
  - Otras informaciones: protección, compartido…

- **Análisis de la segmentación**:
  - Ventajas:
    - No aparece fragmentación interna
    - Facilidad de protección
    - Facilidad para compartir segmentos
    - Desarrollo modular de programas
    - Facilita el enlace dinámico
  - Inconvenientes:
    - Problema de la asignación dinámica de la memoria debido a la diferencia de tamaño de los segmentos
    - Fragmentación externa: aunque hay varios huecos y su suma permitiría alojar el segmento, ninguno por sí solo es lo suficientemente grande como para alojarlo
  - Respecto al tamaño de segmento:
    - Segmentos grandes: aproximación a particiones variables
    - Segmentos pequeños: eliminaría la fragmentación externa pero aumentaría el tamaño destinado a registros
    - Segmentos de tamaño fijo: nos encontraríamos en un esquema de paginación.

4.2.4 Segmentación paginada

- Esta técnica se utiliza cuando los segmentos son grandes. Consiste en paginar los segmentos, de modo que a cada segmento se le asocia su propia tabla de páginas.
- Análisis de la segmentación paginada:
  - Ventajas:
    - Las de la paginación y la segmentación
    - Proporciona protección
  - Inconvenientes:
    - Fragmentación interna en la última página del segmento

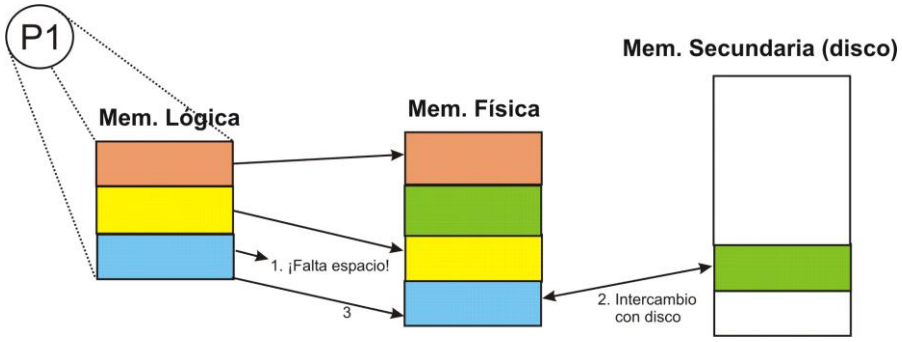
5. Problema de escasez de memoria. Memoria virtual

Las **causas** que provocan este problema son:
- La necesidad de los procesos de una memoria física mayor a la disponible.
- El aumento del grado de multiprogramación.

Técnicas que dan **solución** a este problema son: el método de **intercambios** o **swapping** (utiliza el disco para almacenar la imagen de memoria de los procesos que no caben) y la técnica de **Memoria Virtual**.

La **técnica de Memoria Virtual** permite ejecutar procesos sin que todo su espacio lógico este cargado en memoria principal utilizando la memoria secundaria (disco duro). Consiste en cargar en memoria principal la parte del proceso que se esté ejecutando en ese momento, mientras que la parte del proceso que no se ejecute, se dejará en la memoria secundaria.

Las ventajas que ofrece la memoria virtual son:
- Se puede cargar procesos cuyo tamaño sea superior a la memoria física
- Aumento del grado de multiprogramación
- Ahorro de memoria



Inconvenientes:
- Puede reducir el rendimiento del sistema si no se diseña y sincroniza adecuadamente
- Mayor complejidad

A continuación, se explican las técnicas básicas de memoria virtual: paginación y segmentación por demanda

5.1 Paginación por demanda

Se trata de una técnica que combina las técnicas de: paginación e intercambios de páginas entre memoria principal y secundaria.

La paginación por demanda requiere de **3 bits** en los descriptores de página:
- Bit de validez: indica si una página SÍ está mapeada en la memoria física (1) o NO (0).
- Bit de modificación: indica si la página SÍ ha sido accedida para escritura (lo pone la MMU a 1) o no.
- Bit de referencia: indica si la página Sí ha sido accedida para lectura, escritura o ejecución (lo pone la MMU a 1) o no.

Cuando referenciamos una página que tiene su bit de validez a 0, se produce un **fallo de página**.
Ante un fallo de página, las situaciones que se pueden producir son:
- Página en disco: el proceso a seguir sería:
  1. Encontrar la página demandada en disco.
  2. Encontrar un marco libre:
     - Si existe un marco libre, utilizarlo.
     - Si no, utilizar un algoritmo de reemplazo de páginas para elegir una página victima que deberá dejar su marco a la página demandada.
       Si el bit de modificación es 1, escribir la página en disco (page out).

     Actualizar la tabla de páginas (invalidando la victima) y la tabla de marcos.
  3. Leer la página demandada del disco (page in) y ubicarla en el marco libre, actualizando la tabla de páginas y la de marcos libres.
  4. Transferir el control al proceso de usuario, reejecutando la instrucción que provoco el fallo de página.

- Error de acceso: la página no pertenece al espacio de direcciones lógicas, por lo que se aborta el proceso

A continuación, se presentan algunos **algoritmos de reemplazo de páginas**:

a) **Algoritmo FIFO (**First In First Out): Elige como víctima la página que hace más tiempo que fue cargada en memoria.
Implementación
- Mantiene una cola FIFO de páginas en memoria. Pone al final de la cola la página que se introduce en memoria
- Reemplaza la primera

Ventajas
- Sencillo de implementar.

Inconvenientes:
- Anomalía de Belady: en ocasiones, se producen más fallos de página al aumentar el número de marcos.
- Poco acierto en la selección de la víctima: páginas referenciadas frecuentemente son reemplazadas por ser cargadas en 1er lugar

b) **Algoritmo óptimo:** Reemplaza la página que tardara más tiempo en ser referenciada.
Ventajas:
- Garantiza la mínima tasa de fallos de página posible

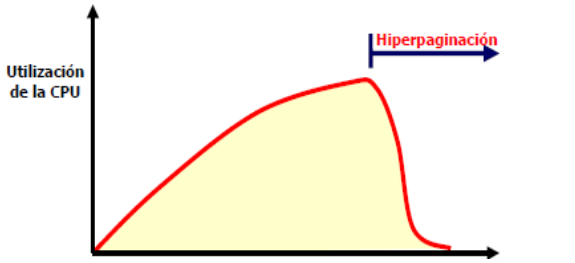
Inconvenientes:
- Implementación imposible: no se puede predecir la página que no será referenciada durante un mayor periodo de tiempo
- Puede servir como referencia para realizar comparaciones.

c) **Algoritmo LRU** (least recently used): Reemplaza la página que hace más tiempo que ha sido referenciada.
Ventajas
- Buena aproximación al algoritmo óptimo.

Inconvenientes:
- Difícil implementación: necesita HW especializado.

Por otro lado, en la asignación de marcos se pueden seguir dos estrategias de reemplazo:
- Estrategia local: un proceso solo puede reemplazar marcos que tenga él asignados.
- Estrategia global: un proceso puede reemplazar marcos del conjunto de marcos total de todos los procesos.

Problema de la paginación por demanda: LA HIPERPAGINACIÓN
Por otro lado, la memoria virtual también puede presentar problemas, como la **hiperpaginación**, la cual se produce cuando un proceso pasa más tiempo salvando y cargando páginas de disco por fallos de página que ejecutándose. Se debe a que el proceso no tiene suficientes marcos para ejecutarse, lo cual produce una serie de fallos en cadena.



5.2 Segmentación por demanda

Es una técnica semejante a la paginación por demanda, pero con segmentos en vez de páginas.
Inconvenientes:
- Es menos eficiente que la paginación, ya que cada vez que se produce un fallo de segmento hay que comprobar si el nuevo segmento cabe en memoria.

Para evitar la fragmentación externa se puede aplicar compactación y si después de aplicarla el segmento continúa sin caber, se procede al reemplazo de segmentos.

6. Gestión de memoria en Linux y en Windows

Linux:
- Utiliza paginación multinivel.
- En el reemplazo de páginas utiliza el algoritmo del reloj (similar al LRU).
- Utiliza una partición de intercambio llamada **swap**.

Windows con núcleo NT (Windows XP, 7, 10…):
- Utiliza paginación multinivel.
- Utiliza un archivo de paginación de disco llamado **pagefile.sys**, cuyo tamaño se fija en panel de control→sistema→rendimiento.

C. GESTIÓN DE ENTRADA/SALIDA

1. Introducción

La **gestión de entrada/salida** es una de las funciones más importantes del SO, ya que el SO debe ser capaz de **manejar los diferentes periféricos de entrada y/o salida existentes**.

¿Por qué es tan importante la gestión de entrada/salida? Porque sin datos de entrada no hay procesamiento, sin dispositivos de salida no hay forma de ver resultados de procesamiento y sin almacenamiento no hay forma de guardar datos o acceder a datos guardados.

De este modo, para gestionar la E/S el SO **debe**:

- Enviar órdenes a los dispositivos de E/S
- Determinar el dispositivo que necesita la atención del procesador
- Detectar las interrupciones
- Controlar los errores
- Proporcionar una interfaz entre los dispositivos y el resto del sistema. Esta interfaz debe ser:
    - Sencilla y fácil de usar.
    - Debe ser la misma para todos los dispositivos.

El SO tiene varias **maneras de llevar a cabo la E/S**, es decir, cuando un proceso necesita acceder a un dispositivo de entrada/salida esto se puede hacer de 3 formas:

- **E/S programada**: mientras se realiza la operación de E/S el procesador está todo el tiempo comprobando si ya ha terminado la operación (por lo que no está haciendo otras funciones más útiles).
- **E/S controlada por interrupciones**: La CPU puede hacer otras actividades mientras se realiza la operación de E/S, ya que se enterará de que se ha completado cuando se produzca una interrupción.
- **E/S mediante el uso de DMA (acceso directo a memoria):** un chip se encarga de la transferencia y accede a la memoria para leer o escribir datos que recibe y envía el dispositivo sin pasar por el procesador.
  Actualmente los discos duros, unidades de CD, DVD, Blueray, admiten **DMA** y la tienen activada por defecto.

Hay otro **problema** al cual el Sistema Operativo tiene que hacer frente y es el tema de las **velocidades**. Cada dispositivo lee y escribe datos a una velocidad diferente, por ejemplo, el teclado escribe datos de forma más lenta que la impresora. Además, todos los dispositivos de entrada/salida son mucho más lentos que el microprocesador. Por este motivo se utilizan **técnicas de almacenamiento intermedio** para mejorar el rendimiento del sistema:

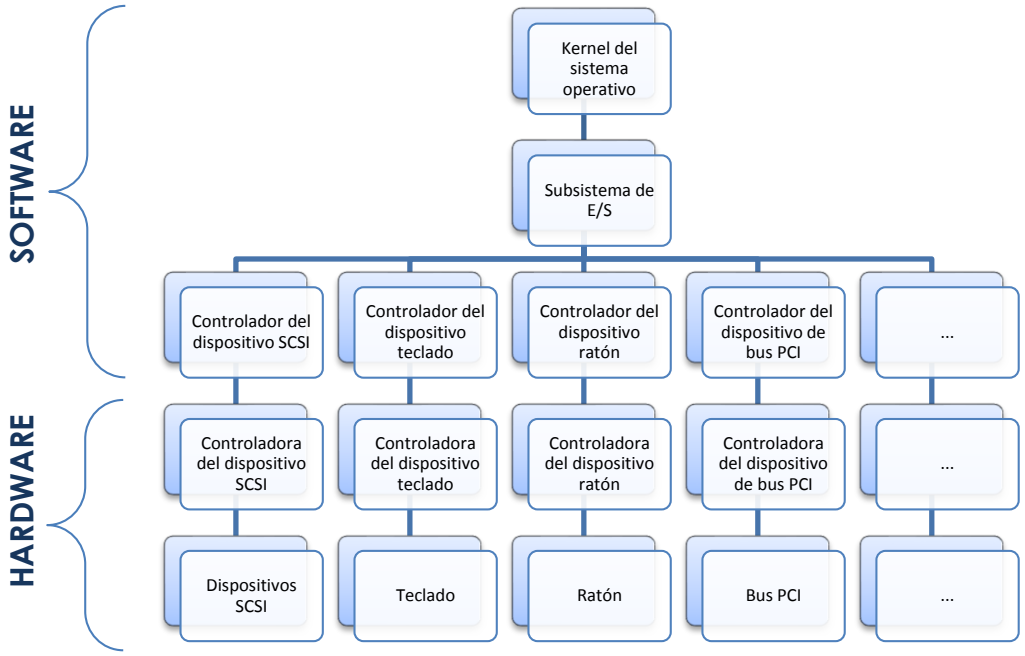
- **Caching**: consiste en almacenar una caché temporal, de rápido acceso, los datos que se usan con más frecuencia.
- **Buffering:** es una de las técnicas para evitar sobrecargas e ineficiencias en las operaciones de E/S. Consiste en utilizar un área de memoria como buffer, simulando un dispositivo o un periférico lógico, que hará de dispositivo intermedio entre el periférico real y el procesador. El buffer es independiente del dispositivo de entrada y/o salida, por lo que permite que el procesador comience a trabajar leyendo o almacenando en el buffer mientras la información del periférico se va almacenando o extrayendo del buffer. Esto evita que un periférico lento afecte al rendimiento del equipo informático.

  Un ejemplo se puede observar cuando se está visualizando un contenido multimedia.
- **Spooling**: técnica en la cual la computadora introduce trabajos en un buffer (un área especial en RAM o un disco), de manera que un dispositivo pueda acceder a ellos cuando esté listo. El spooling es útil en caso de dispositivos que acceden a los datos a distintas velocidades. El buffer proporciona un lugar de espera donde los datos pueden estar hasta que el dispositivo (generalmente más lento) los procesa. Esto permite que la CPU pueda trabajar en otras tareas mientras que espera que el dispositivo más lento acabe de procesar el trabajo.

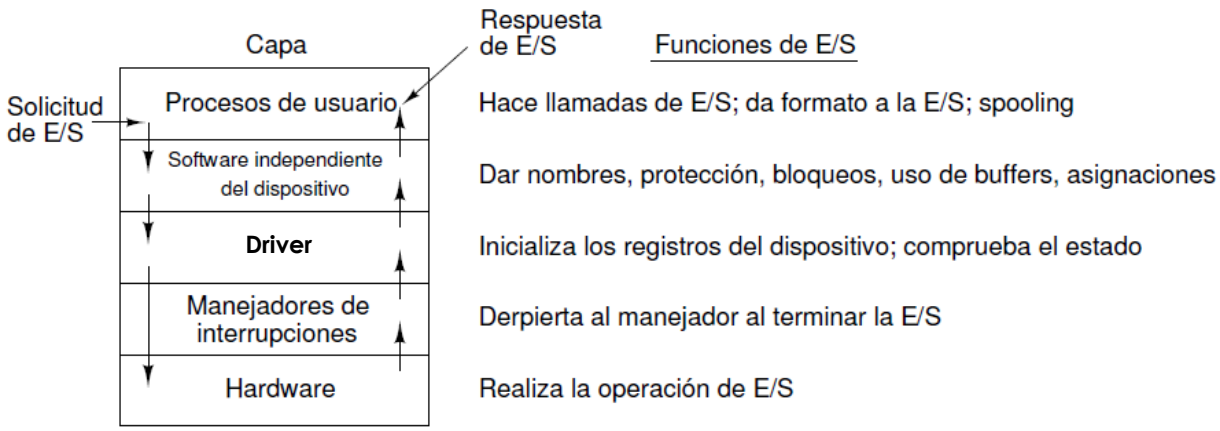
La aplicación más común del spooling es la impresión. En este caso, los documentos son cargados en un área de un disco, y la impresora los saca de éste a su propia velocidad. El usuario puede entonces realizar otras operaciones en el ordenador mientras la impresión tiene lugar en segundo plano. El spooling permite también que los usuarios coloquen varios trabajos de impresión en una cola de una vez, en lugar de esperar a que cada uno acabe para enviar el siguiente.

2. Software de entrada/salida

Estructura del módulo de E/S en un sistema operativo:

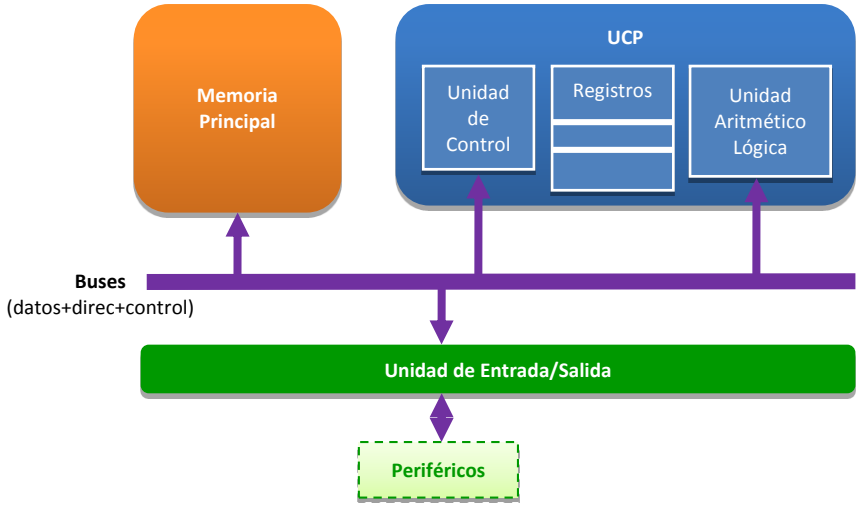


El software de E/S se organiza en **niveles**. Los del nivel inferior ocultan las particularidades del hardware a los del nivel superior que presentan una interfaz simple y uniforme al usuario.



3. Hardware de entrada/salida

3.1. Unidad de entrada/salida (chipset)



La Unidad de Entrada/Salida (chipset) permite la comunicación de la UCP y la Memoria Principal con el exterior: impresoras, monitor, teclado, etc.

Para que se pueda llevar a cabo el intercambio de información se deben realizar las siguientes tareas:

- **Direccionamiento**: selección del dispositivo de E/S implicado en una transferencia determinada
- **Sincronización** de UCP y periféricos: es necesario coordinar la actividad de la UCP con los periféricos, ya que sus velocidades de trabajo son distintas.
- **Transferencia** de los datos desde o hacia el dispositivo seleccionado
- **Control de errores.**

Actualmente la unidad de entrada/salida posee su propia memoria local y es, de hecho, un pequeño ordenador independiente. La CPU tiene una participación mínima. Solo se intercambian mensajes.

3.2. Buses

Conjunto de líneas conductoras que permiten la comunicación entre los dispositivos y el resto de componentes del sistema.
Algunos buses pueden conectar varios dispositivos.

3.3. Puertos ("dirección" del dispositivo)

Punto de conexión entre el dispositivo de E/S y el resto de componentes del sistema

3.4. Dispositivos periféricos

Partes:

En general, los periféricos constan de:

- Un componente **electrónico** (circuito impreso), denominado **controladora**
- Un componente **mecánico** que es el **dispositivo** mismo.

Controladora:

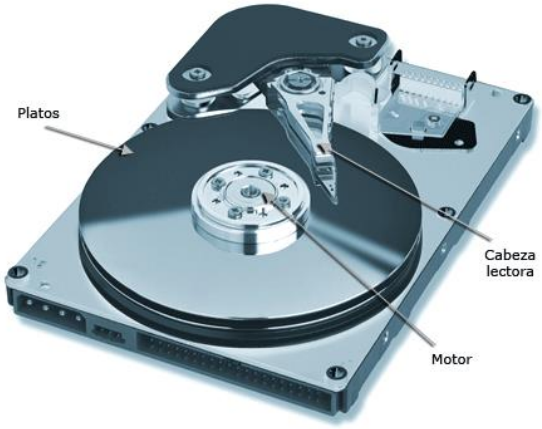
- Es un HW encargado de conectar físicamente el periférico a la placa base.
- Necesita de un SW (driver o controlador).
- Las principales funciones de las controladoras son:
  - Comunicación con el periférico, intercambio de órdenes, información del estado.
  - Detección de errores.
  - Comunicación con el procesador, descodificadores de órdenes, datos, información de estado, reconocimiento de dirección.

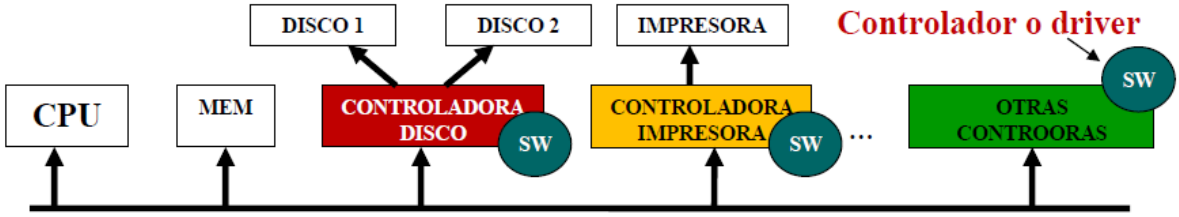
- Al código o sw específico que el SO utiliza para comunicarse con una controladora se lo conoce como **driver** (también llamado manejador o controlador) de la controladora. De este modo, para llevar a cabo las tareas de E/S, el SO, usando el driver, se comunica con la controladora a través de una serie de registros específicos que cada controladora tiene. Cuando la orden ha sido cumplida, la controladora produce una interrupción con el fin de permitir que la UCP atienda al SO para comprobar los resultados de la operación de E/S. Para dicha comprobación se utilizan los valores de los registros de la controladora que informan sobre el estado final.

- Por ejemplo, en el caso de un disco duro, la controladora sería el circuito con chips que hay en la parte inferior.

Dispositivo (parte mecánica):

- Es el dispositivo en sí
- Por ejemplo, en el caso de un disco duro sería sus platos, motores, cabezales…





Tipos de periféricos

Los periféricos se pueden clasificar en función de si gestionan la información por **bloques** (leen o escriben bloques de información de un tamaño fijo, los cuales tienen su propia dirección y pueden ser accedidos de forma independiente. Ej: discos duros) o por **caracteres** (reciben o envían la información en forma de flujo de caracteres. Ej: teclado, impresora).

También podemos clasificar los periféricos en:

- De **entrada** de datos: teclados, ratón,…
- De **salida** de datos: pantalla, impresora, altavoces,…
- De **entrada y salida** de datos: pantalla táctil, disco duro,…

4. Gestión de discos

Los discos son almacenamiento masivo no volátil. **Dan soporte al sistema de archivos y al sistema de memoria virtual**.

Principales ventajas de los discos:

- Mayor capacidad de almacenamiento.
- Menor precio por bit.
- La información no se pierde al apagar el PC.

Particiones en un disco duro:

Se pueden tener diferentes tipos de sistemas de archivos particionando un disco duro, es decir, dividiendo el disco físico en varios discos lógicos o particiones.

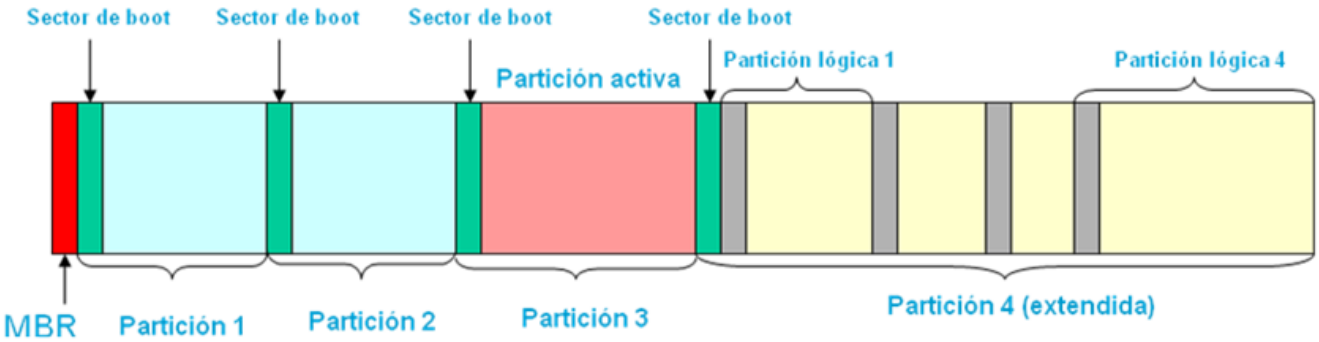
Existen dos formas de gestionar estas particiones: MBR y GPT.

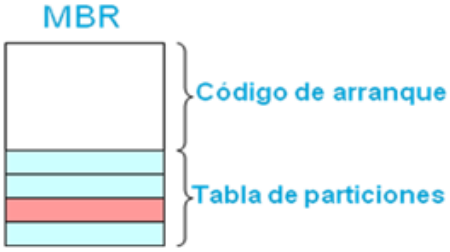
MBR

El MBR (Master Boot Record o registro de arranque maestro) es el primer sector de un disco duro y contiene la información sobre las particiones de la unidad. De este modo, cuando un ordenador finaliza su inicialización a través del proceso BIOS, entonces comprueba el MBR para obtener la información del arranque del sistema. En función de los datos que tenga el MBR, el equipo cargará el sistema operativo o presentará una lista de arranque de sistemas operativos.

Principales características de MBR:

- Sólo soporta hasta 4 particiones primarias, o hasta 3 primarias y una extendida (con hasta 128 lógicas en la extendida). Para poder tener más de cuatro particiones en un disco, existe un tipo especial de partición, la partición extendida. Una partición extendida permite definir hasta 128 particiones lógicas dentro de ella.
- Soporta todas las máquinas de 32 y 64 bits
- Soporta hasta 2TB por partición
- Usa la BIOS estándar
- Es soportado por la gran mayoría de SO

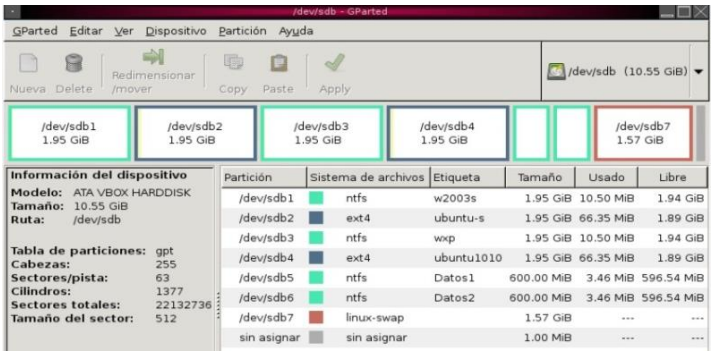




- El MBR (Master Boot Record) ocupa el primer sector del disco, conteniendo el programa de arranque y la tabla de particiones con cuatro entradas.
- Cada entrada de la tabla contiene el tipo de partición, si es la partición activa o no y la posición de inicio y fin en el disco

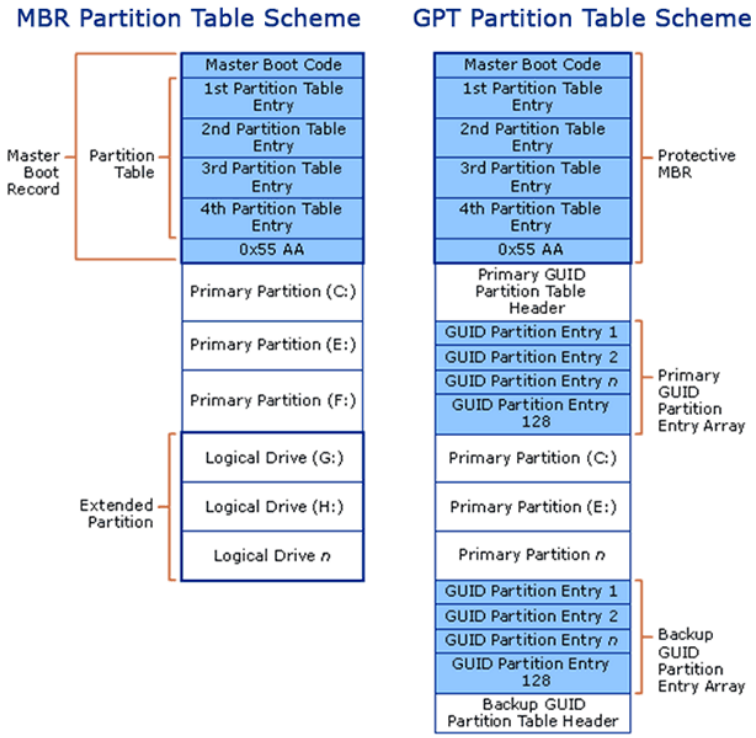
GPT:

La tabla de particiones GUID (GPT) se introdujo como parte del Firmware Extensible Unificado Interface (UEFI). GPT proporciona un mecanismo más flexible para particionar los discos de arranque que el Master Boot Record (MBR). Este nuevo sistema fue creado para mejorar las limitaciones que planteaba MBR.

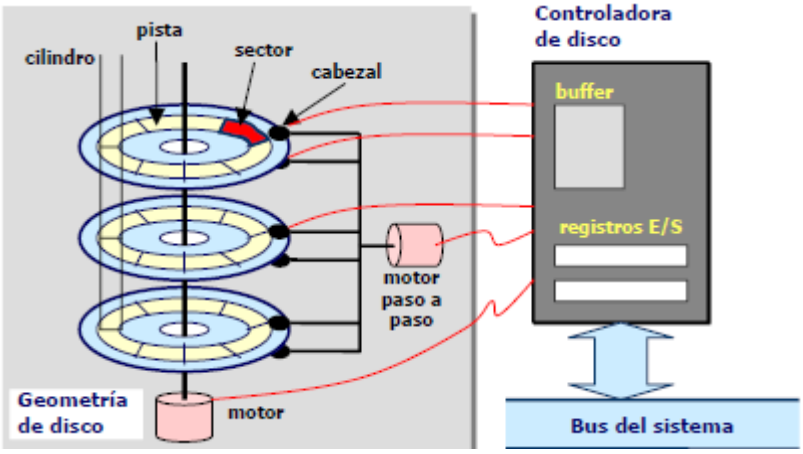


Principales características de GPT:

- Soporta hasta 128 particiones primarias.
- Soporta sólo máquinas de 64 bits.
- Soporta hasta 256TB por partición.
- Funciona con EFI
- Es soportado por SO con arquitectura de 64 bits



Estructura de un disco:



Cilindro

Pista: Las pistas (tracks) son los círculos concéntricos en los que se divide cada cara

Sector o bloque físico (Nota: Un clúster es un conjunto contiguo de sectores que componen la unidad más pequeña de almacenamiento de un disco. Los archivos se almacenan en uno o varios clústers dependiendo de su tamaño)

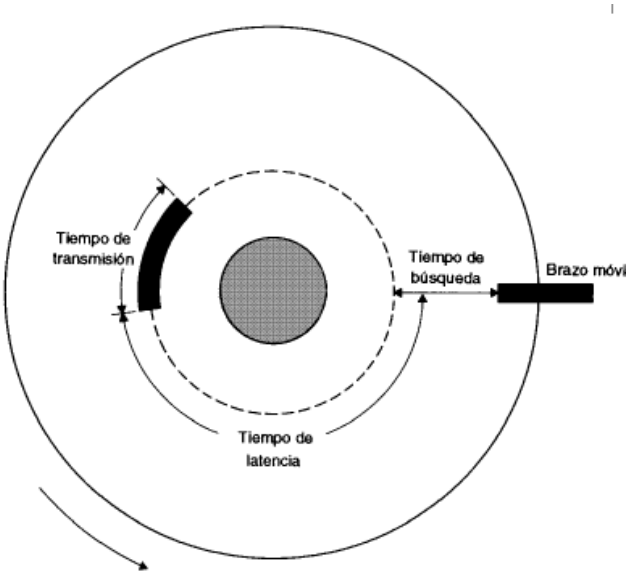
Cabezal

Vista interior disco duro: http://www.youtube.com/watch?v=Y7U8M6UsEwE

El tiempo de leer o escribir un sector del disco está determinado principalmente por estos factores:

- Mover los cabezales sobre el cilindro solicitado => tiempo de **búsqueda**
- Esperar a que el sector pase ante la cabeza => tiempo de **latencia**
- Leer o escribir el sector => tiempo de **transmisión**

El más costoso es el tiempo de **búsqueda** (tiempo que tarda el disco en mover los cabezales sobre el cilindro solicitado), por lo que las peticiones a disco deben ser **planificadas** de modo que se siga un orden que **minimice el recorrido del cabezal** (siguiendo algoritmos como el FCFS, SSTF, SCAN…). Esto tiene que ser logrado por el manejador del dispositivo (**driver**).

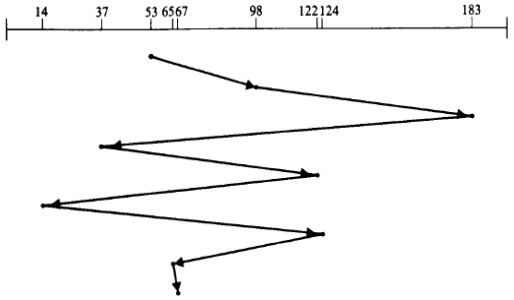


Algoritmos de planificación:

FCFS (primero en llegar – primero en ser servido)

- Planificación por orden de llegada.
- Ventajas: implementación simple
- Inconvenientes: recorridos totales elevados y movimientos bruscos del cabezal

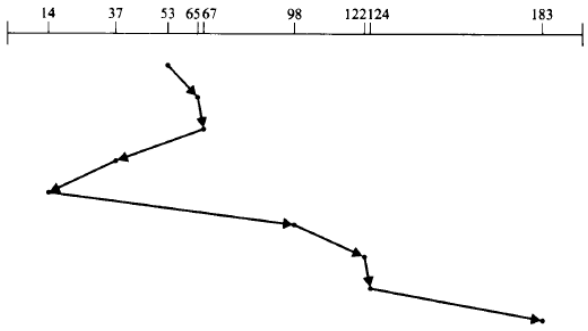
Ejemplo: Posición cabezal: cilindro 53
Cola (nº de cilindro): 98, 183, 37, 122, 14, 124, 65, 67



SSTF (menor tiempo de búsqueda primero)

- Primero, el cilindro más cercano (menor tiempo de posicionamiento)
- Ventajas: recorridos totales más bajos
- Inconvenientes: El cabezal oscila sobre la zona central (posible inanición)

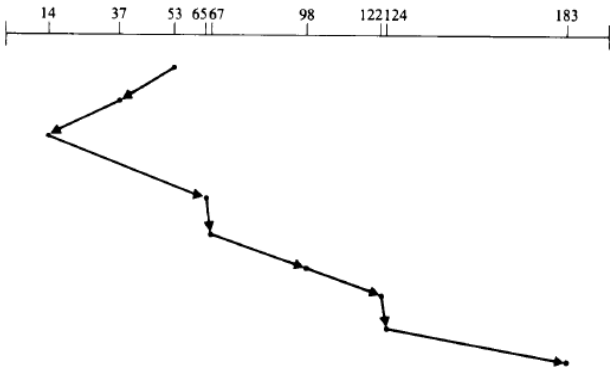
Ejemplo: Posición cabezal: cilindro 53
Cola (nº de cilindro): 98, 183, 37, 122, 14, 124, 65, 67



SCAN (barrido)

- El cabezal recorre el disco de un extremo a otro atendiendo las peticiones que encuentra en su camino; cuando llega al final cambia de dirección.
- Ventajas: recorrido total bajo, sin inanición
- Inconvenientes: las peticiones de los extremos tienen tiempos de servicio altos.

Ejemplo: Posición cabezal: cilindro 53
Cola (nº de cilindro): 98, 183, 37, 122, 14, 124, 65, 67



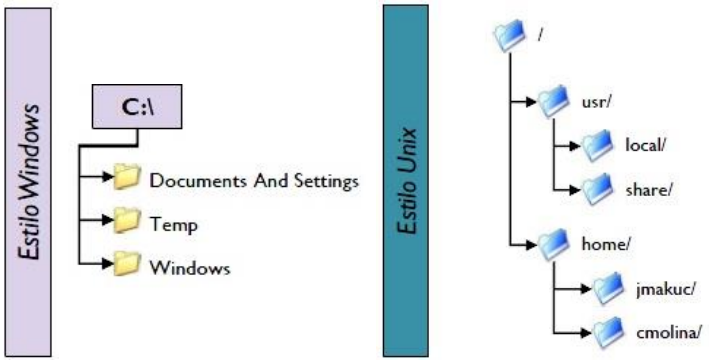
D. GESTIÓN DE FICHEROS

1. Introducción

- **Fichero o archivo**: Conjunto de información de un determinado tipo que esta almacenada en un dispositivo de almacenamiento. Ejemplo: documento de texto, canción, imagen...
- **Carpeta o directorio:** Tipo especial de fichero que se utiliza para organizar ficheros (u otras carpetas).
- **Sistema de ficheros:** Parte del SO que permite "administrar" la información almacenada de los dispositivos de E/S en forma de ficheros.

  Sus objetivos son:

  - Crear, modificar o borrar ficheros (o carpetas)
  - Controlar el acceso a los ficheros (mediante permisos)
  - Permitir intercambio de datos entre ficheros.
  - Permitir realizar copias de seguridad de los ficheros
    - Permitir el acceso a los ficheros mediante nombres simbólicos.



2. Ficheros

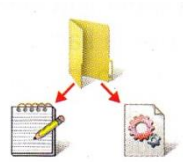
El sistema de ficheros determina algunos atributos y propiedades de los ficheros

- **Nombre y extensión de los ficheros**
  - **Nombre**: La mayoría de SO permiten usar nombres de hasta 255 caracteres y algunos SO, como Linux, distinguen entre mayúsculas y minúsculas.
  - **Extensión**: sirve para saber el programa que permite ejecutar o abrir un fichero. Algunos SO como Linux no necesitan el uso de extensiones.
- **Tipos de ficheros:**
  - Ficheros normales o regulares: Aquellos ficheros que contienen datos (información).
  - Directorios: fichero que se utiliza para organizar los ficheros (u otras carpetas).
  - Ficheros especiales de dispositivos: representan a dispositivos de E/S
- **Información que contiene un fichero:**
  - Nombre
  - Tamaño
  - Fechas: de creación, modificación…
  - Propietario
  - Permisos (lectura, escritura, ejecución...)
  - Ubicación
  - Enlaces: puntos desde los que se puede acceder al fichero.
- **Operaciones que se puede hacer sobre un fichero:**
  - Crearlo
  - Abrirlo
  - Escribirlo
  - Cerrarlo
  - Borrarlo

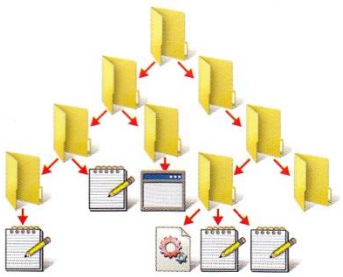
3. Directorios

- **Operaciones que se puede hacer sobre un fichero:**
    - Crearlo
    - Entrar en él
    - Salir de él
    - Leer su contenido
    - Añadir o eliminar en él archivos o directorios
    - Borrarlo

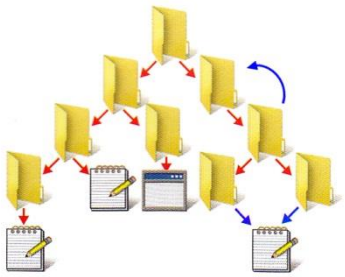
- **Estructuras de directorios**:
  Tipos:
    - **Un nivel**: Todos los ficheros en un único directorio



    - **Jerárquico**: un directorio puede incluir otros directorios y archivos



Árbol de directorios



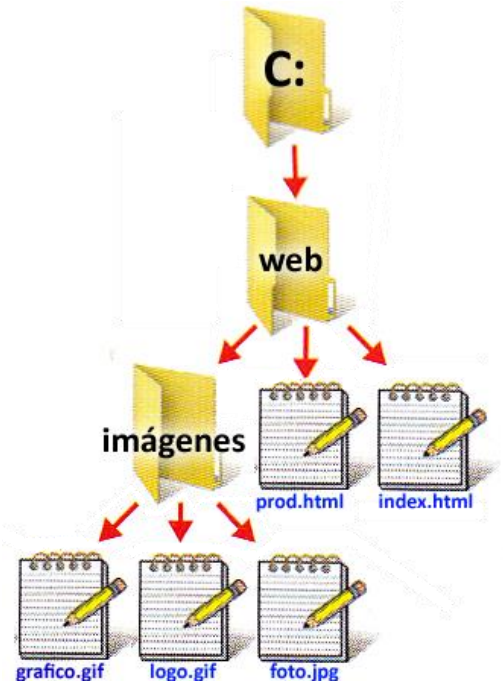
Grafo de directorios

La mayoría de los sistemas operativos tienen un sistema de archivos de **estructura jerárquica**, en el que los directorios parten de uno llamado directorio raíz, y del que cuelgan todos los demás en forma de árbol, de ahí que se utilicen términos como árbol de subdirectorios.

- **Directorios especiales**:
    - Existen dos tipos:
        - . → directorio actual
        - .. → directorio padre

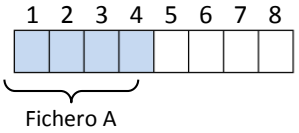
- **Rutas**: concatenación de directorios y subdirectorios para llamar a un archivo en una estructura de directorios.

  Tipos:
    - **Absolutas**: se llama al archivo desde el directorio raíz hasta el archivo. Ejemplo: C:\web\imagenes\logo.gif
    - **Relativa**: se llama al archivo desde el directorio actual en el que estemos. Ejemplo: si estamos en la carpeta "web" la ruta hasta llegar al archivo "logo.gif" sería: imagenes/logo.gif

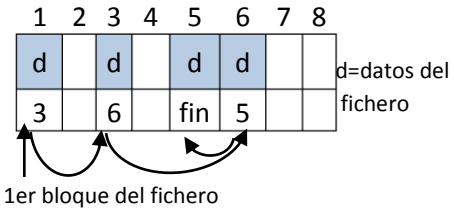


4. Métodos de asignación

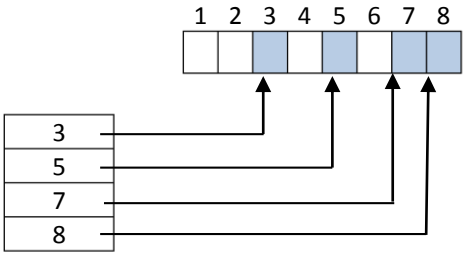
- **Definición**: métodos de asignar espacio a cada fichero dentro del disco.



- **Tipos**:
  - Asignación **contigua**: los bloques de un fichero se encuentran de forma contigua en el disco.
  - Asignación **enlazada**: en cada bloque está parte de los datos del fichero y una pequeña parte para indicar el siguiente bloque que contiene los datos del fichero.



  - Asignación **indexada**: para cada fichero, el SO tiene una pequeña tabla de índices que indican los bloques de cada fichero.



5. Ejemplos de sistemas de ficheros

	FAT16	FAT32	NTFS	ext4
Sistemas Operativos	MS-DOS 6.22 W95 W98	W98SE W2000 WXP W2003 W2008 W7	W2000/2003/2008/2012/2016/2019/2022 Server WXP Vista W7 W8 W10	Linux
Máx. Tamaño Archivo	2 GB	4 GB	16 TB	16TB
Máx. Tamaño Partición	2 GB	2 TB	256 TB	1EB

FAT16:

- El tamaño de las particiones y de los archivos no podía ser superior a 2GB
- El nombre de los ficheros tenía que ser de 8 caracteres para el nombre y 3 caracteres para la extensión.

FAT32:

- Este sistema de archivos se introdujo para el SO Windows 95 y fue usado por Win98 y Me.
- Soporta particiones de 2TB
- El nombre de los archivos puede tener más caracteres (255) y los archivos pueden ser de mayor tamaño (4GB).

NTFS:

- Mejora el sistema de ficheros anterior introduciendo mayor seguridad, mayor estabilidad, compresión y encriptación nativa, y mayor tamaño de los archivos.
- Se utiliza para los SO Windows NT, 2000, XP, Vista, 7, 8 y los Server 2000/2003/2008/2012/2016/2019/2022.

EXT2, EXT3, EXT4:

- Sistemas de archivos soportados por la mayoría de las distribuciones Linux.
- La versión ext3 es una mejora de la ext2, con algunas características nuevas como el journaling (técnica que registra a diario los cambios en el sistema de archivos para poder recuperar los datos en caso de fallo) y el ext4 es una mejora de ext3, con más mejoras como soportar particiones y ficheros más grandes, mejora de la fiabilidad y la rapidez.
- Los tres sistemas son compatibles entre sí.

OTROS:

- Otros sistemas de archivos en Linux son XTS o reiserFS
- En Mac OS destacamos los sistemas de archivos Mac OS Extended (HFS Plus) que reemplaza a HFS y el sistema de archivos UFS.
- En UNIX destacamos UFS.

6. Gestión de ficheros en Linux y en Windows

Windows:

- Estructura de directorios: árbol de directorios
- Método de asignación:
  - FAT: asignación enlazada
  - NTFS: es un hibrido entre contigua y no contigua. Si se puede, el archivo utiliza un conjunto contiguo de bloques de datos (igual que la asignación contigua). Si no basta con un espacio contiguo, se busca otra extensión de bloques contiguos. En la entrada en directorio se guarda la información de todas las extensiones. Por cada extensión: bloque inicial, nº bloques.

Linux:

- Estructura de directorios: grafo de directorios
- Método de asignación: asignación indexada

EVOLUCIÓN HISTÓRICA DE LOS SISTEMAS OPERATIVOS

1ª generación (1940-1959)	El Mark I y el Eniac no tenían todavía SO (se programaban con interruptores mecánicos) GM-NAA I/O del IBM 704 de General Motors (1956): es hoy considerado el primer SO
2ª generación (1960-1965)	SO de procesamiento por lotes o batch EXEC I y EXEC II (programado en cobol) del Univac 1107 (1962): SO basado en procesamiento por lotes CTSS desarrollado por el MIT (1961): SO de tiempo compartido EXEC 8 del Univac 1108 (1964): de tiempo compartido, multiproceso, tiempo real, usaba Fieldata y con sistemas de archivos que permitía compartir directorios y proteger ficheros
3ª generación (1966-1971)	Los SO eran monousuarios OS/360: SO de los macroordenadores o mainframes de la serie 360 de IBM CP/CMS y VM/CMS (principios de los 70's): usados en la familia de ordenadores de IBM 370 TOPS-10: usado en los ordenadores de la serie PDP de la empresa DEC MULTICS del macroordenador GE 645 (mediados 60's): SO monolítico, fue la base para el diseño de UNIX UNIX (1970): usado en el PDP-11
4ª generación (1971-1981)	SO multiusuario y multitarea Interfaz de usuario sigue siendo modo texto TOPS-20 UNIX (principio 70's): programado en C, multiusuario, multitarea, derivado de Multics, permitía compartición de recursos y uso remoto de equipos. BSD UNIX (finales 70's): variante de UNIX CP/M (1976): muy utilizado, fiable y fácil de manejar VMS del VAX de la empresa DEC: multiusuario y multiproceado
5ª generación (1981-…)	PC-DOS del IBM-PC: basado en el CP/M XENIX de Microsoft: versión de UNIX para PCs MS-DOS de Microsoft: interfaz en modo texto Windows 1.0, 2.0, 2.1, 3.0, 3.1, 3.11 en realidad no eran SO, eran programas instalados sobre MS-DOS Windows 95: ya se instalaba independientemente de MS-DOS Windows 98, ME, XP, Vista, 7, 8 y 10 Novell Netware: SO de red Windows NT, 2000/2003/2008/2012/2016/2019/2022 Server: SO en red de Microsoft OS 400 del ordenador AS 400 de IBM Linux (principios 90's): versión de UNIX para PC's, SO libre y de código abierto Solaris de Sun (1992): versión de Unix que ha liberado para del código fuente Mac OS del Mac de Apple

BIBLIOGRAFÍA

- Jiménez, I. "Sistemas Informáticos". Garceta. 2012.
- Stallings, W. "Sistemas Operativos". Pearson. 2005.