# Software de un Sistema Informático

## Índice:



3.- Gestión de procesos.
- 3.1.- Planificación del procesador.
- 3.2.- Planificación apropiativa y no apropiativa.
- 3.3.- Cambio de Contexto

4.- Gestión de memoria.
- 4.1.- Gestión de memoria en sistemas operativos monotarea.
- 4.2.- Gestión de memoria en sistemas operativos multitarea.
- 4.2.1.- Asignación de particiones fijas.
- 4.2.2.- Asignación de particiones variables.
- 4.2.3.- Memoria virtual - Técnicas de memoria virtual.

5.- Gestión de la entrada/salida.
- 5.1.- Controladores de dispositivo.
- 5.2.- Estructura de datos de la E/S.
- 5.3.- Técnicas de la E/S.
- 5.4.- Planificación de discos.

6.- Gestión del sistema de archivos.
- 6.1.- Organización lógica y física.
- 6.2.- Operaciones soportadas por un sistema de archivos.
- 6.3.- Rutas de acceso.




# 1.- Software de un sistema informático.

Definimos el concepto de sistema informático como un conjunto de elementos que hacen posible el **tratamiento automatizado de la información**.

Nos vamos a centrar en el **software de un sistema informático**.

Formado por:

- **Programas: Sistema Operativo - Aplicaciones**
- **Datos**
- **Documentación** asociada.

El software está distribuido entre ordenador, periféricos y subsistema de comunicaciones.

***Ejemplos** de software son:*

- *los sistemas operativos*
- *paquetes ofimáticos, compresores, editores de imágenes*
- *paquetes de programación*
- *controladores de dispositivos*
- *nuestros documentos*

En todo **proceso de instalación** de una aplicación software se han de seguir los siguientes **pasos**:

- Determinación del **equipo necesario**.
- **Ejecución** del programa de instalación.
- **Configuración** de la aplicación.

## 1.1.- Requisitos e instalación: Determinación del equipo necesario.

Conocer **qué necesita la aplicación** para que funcione adecuadamente en el ordenador, es decir, qué **características** o **requisitos** tendrá que tener el sistema informático.

Cada desarrollador crea sus aplicaciones enfocadas a:

- **plataformas concretas** (no podrá ser instalada en otra distinta)
- **necesidades de hardware/software necesarias** para su funcionamiento (sistema informático debe cumplir unos requisitos mínimos).

Tendrás que reunir la información sobre el hardware de tu ordenador y deberás verificar que tu hardware te permite realizar el tipo de instalación que deseas efectuar.

Actividad: Indica ejemplos de requisitos que consideres que se deben tener en cuenta antes de instalar un software

*Ejemplo de Requisitos para instalar software:*

- ***Plataforma** hardware: PC, Mac, etc.*
- ***Procesador**: fabricante, velocidad, generalmente se indica el inferior posible de la gama con el que la aplicación funciona adecuadamente.*
- *Memoria **RAM** mínima.*
- ***Espacio** mínimo disponible en el soporte de almacenamiento (disco duro o unidad de almacenamiento externa para aplicaciones portables).*
- *Tarjeta gráfica: la **memoria gráfica** necesaria para el buen funcionamiento de la aplicación.*
- ***Resolución** recomendada del monitor.*
- *Plataforma software: **sistema operativo** bajo el que funciona la aplicación, Windows, Linux, etc.*
- *Otros paquetes **software adicionales** necesarios:*
  - *actualizaciones concretas de seguridad para el sistema operativo*
  - *JVM (máquina virtual de Java).*

Los fabricantes de aplicaciones informáticas suelen establecer **tres niveles de requisitos** para la instalación de sus aplicaciones:

- Equipo **básico**.
- Equipo **opcional/recomendado**.
- Equipo en **red**.

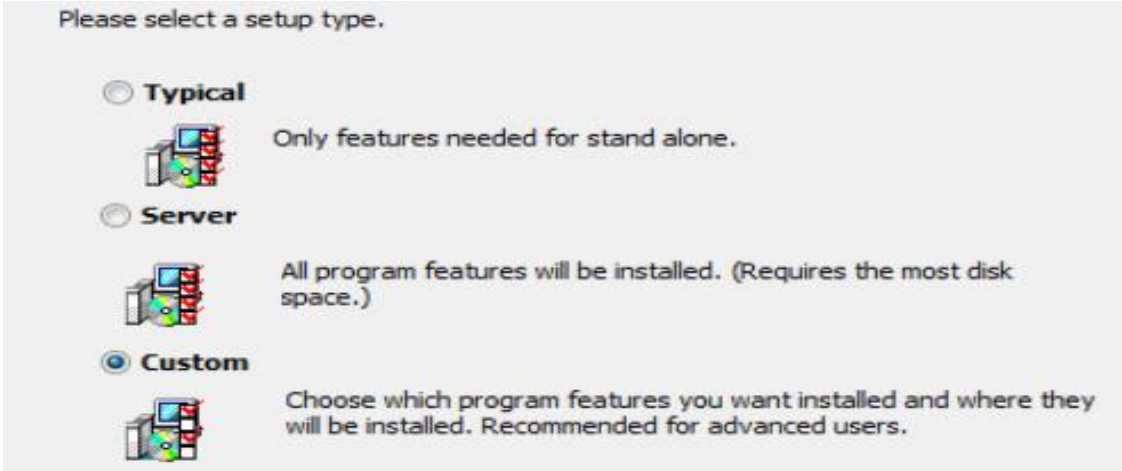
## 1.2.- Requisitos e instalación: Ejecución del programa de instalación.

Instalación de un programa es el **conjunto de pasos que nos permite:**

- **copiar los archivos necesarios**
- **poner en funcionamiento** la aplicación en un sistema informático
- **configurar** la aplicación

La mayoría de las aplicaciones presentan **dos niveles** en función del usuario:

- Instalación **básica:** para usuarios con pocos conocimientos informáticos. Realizará la instalación en función de los elementos que detecte en el equipo y según unos parámetros básicos establecidos por defecto por el fabricante.
- Instalación **personalizada** o **avanzada:** usuario experto, permite incluir o eliminar elementos de la aplicación con el fin de **optimizar** los recursos del sistema informático, instalando sólo aquellos elementos de la aplicación que se van a utilizar.



Cuando se adquiere una aplicación informática, nos encontramos con:

- un grupo de **manuales.**
- un **conjunto de ficheros o fichero** que se encuentra normalmente en formato **comprimido** en algún tipo de **soporte**.

El **traspaso(instalación)** del programa al **soporte de almacenamiento** de nuestro ordenador (disco duro), se realiza a través del **programa de instalación** (setup, install, etc.).
Es el encargado:

- de **extraer** la aplicación, **descomprimirla.**
- crear la **estructura de directorios** necesaria.
- ubicar los **archivos de la configuración** donde corresponda.
- si fuera necesario, modificar el **registro** del sistema.

En la actualidad, los fabricantes distribuyen sus aplicaciones con posibilidad de **descarga** de los archivos de instalación o en **imágenes ISO** (Ejemplo: muchas distribuciones de Linux pueden descargarse en este formato).

## 1.3.- Requisitos e instalación: Configuración de la aplicación.

Configurar las **opciones** de la aplicación, del Sistema Operativo (registro), y del entorno de trabajo. Permite modificar los parámetros establecidos por **defecto** para la aplicación.

Las aplicaciones generan una serie de **archivos de configuración** con los datos introducidos por los usuarios, podemos realizar copias de seguridad para reinstalaciones futuras.

La configuración del **entorno de trabajo** consiste en definir una serie de parámetros de funcionamiento (salvo que no sea satisfactoria la configuración establecida por defecto por el programa), que adecuen el funcionamiento de la aplicación a las **exigencias del usuario**.
Estos parámetros pueden ser:

- Ajuste de la **pantalla** (tamaños de las ventanas, colores, tipos de letras, resolución, etc.).
- Definición de **directorios de trabajo** (donde guardar los archivos, proyectos, plantillas, etc.).

*Ejemplo de configuración aplicación server para control remoto VNC: cambiar la **contraseña** de administrador, cambiar los **puertos** por defecto, etc.*
*Ejemplo aplicaciones web: requieren la activación de **cookies** y la modificación de la configuración de **seguridad** de nuestro navegador.*

***¿Qué nivel de requisitos en la instalación de una aplicación recomienda el fabricante software para conseguir un rendimiento <u>óptimo</u>?***

- *Requisitos del equipo opcional.*
- *Requisitos del equipo en red.*
- *Requisitos de compatibilidad.*
- *Requisitos del equipo básico.*

## 1.4.- Tipos de Software.

Clasificamos las aplicaciones informáticas en **tres tipos** en función de la **naturaleza de uso**:

- Software del **Sistema**: para gestionar y controlar los componentes de hardware. Se encargará del buen funcionamiento de todos los componentes, herramientas para acelerar el equipo.
- Software de **Aplicaciones**: completar una o varias tareas:
  - Aplicaciones de **propósito <u>general</u>**.
  - Aplicaciones de **propósito <u>específico</u>**.
- Software de **Programación**: permite a los desarrolladores de aplicaciones utilizar lenguajes de programación para crear, mantener y ejecutar programas.
  - Lenguajes de Programación
  - Programas de Bibliotecas
  - Frameworks



Actividad: Indica ejemplos de ambos tipos de Aplicaciones.

**Aplicaciones de propósito general (ofimático):**

Para el desempeño de **funciones no específicas:**

- Gestión de **Texto:** se suelen comercializar en paquetes integrados denominados **suites**.
    - **Editores** de texto (no permiten formato, como por ejemplo Notepad).
    - **Procesadores** de texto (Microsoft Word, Writer de OpenOffice).
- **Hoja de Cálculo** (Microsoft Excel, Calc de OpenOffice).
- Gestores de agenda, calendario, contactos.
- Generador de Presentaciones (Microsoft PowerPoint, Impress de OpenOffice).
- **Herramientas para la comunicación**: gestores de **mail**, servicio de **mensajería** instantánea.
- **Utilidades y herramientas**: antivirus, navegadores web, gestores de archivos, etc.

**Aplicaciones de propósito específico (técnico):**

Se utilizan para el desempeño de funciones específicas, científicas, técnicas o de gestión como:

- **Administración**, contabilidad, facturación, gestión de almacén, RRHH (ContaPlus)
- Entorno gráficos de **Desarrollo** (Eclipse)

- Herramientas de administración de **bases de datos**

*Ejemplo: Oracle, phpMyAdmin.*

- Herramientas de **gestión de red**
- Herramientas **especializadas**

*Ejemplo: OCR, gestión empresarial ERP.*

- Herramientas de **diseño gráfico y maquetación**

*Ejemplo Adobe PhotoShop, PaintShop.*

- Herramientas de **ingeniería y científicas** en ámbitos de investigación, en universidades

*Ejemplo: Matlab.*

***Un antivirus y un entorno de desarrollo para programación son ejemplos de:***

- *Aplicaciones de propósito específico.*
- *Aplicaciones de propósito general.*
- *Aplicaciones de propósito específico y general, respectivamente.*
- *Aplicaciones de propósito general y específico, respectivamente.*


Otra categoría son las **aplicaciones portables**. Puedes llevarlas en tu memoria USB y utilizarlas donde y cuando quieras, sin necesidad de instalación. Para ello, visita este enlace:

https://portableapps.com/es

Actividad – Trabajo sobre Aplicaciones Portables

## 1.5.- Licencias software

Una **licencia** software sirve para establecer un **contrato** entre el **autor de una aplicación software** (sometido a propiedad intelectual y a derechos de autor) y el **usuario**.
En el contrato se **definen con precisión los derechos y deberes de ambas partes**, es decir, los "actos de explotación legales".

Entendemos por **derecho de autor o copyright** la forma de protección proporcionada por las leyes vigentes en la mayoría de países para los autores de obras originales (literarias, musicales, artísticas, intelectuales, informáticas, tanto publicadas como pendientes de publicar).
Pueden existir tantas **licencias** como acuerdos concretos se den entre el autor y el usuario.

Algunos **tipos de licencias** en función de lo limitadas que estén las **acciones del usuario:**

- Software **propietario**.
- Software **libre**.
- Software semilibre.
- Software de dominio público.
- Software con **copyleft**.

**Software propietario**

Software cuya redistribución o modificación están **prohibidos** o **necesitan una autorización.**
Los usuarios **tienen limitadas las posibilidades de usarlo, modificarlo o redistribuirlo**.
Su **código fuente no está disponible**, o el acceso a éste se encuentra **restringido**.
Una licencia software propietario lo que **otorga** es el **derecho de uso de la aplicación**.

En el software propietario o "no libre" una persona física o jurídica (compañía, corporación, fundación, etc.) **posee los derechos de autor sobre un software**, **no** otorgando**:**

- **derechos de usar el programa con cualquier propósito.**
- de estudiar cómo funciona el programa y **adaptarlo** a las propias necesidades (el acceso al código fuente es una condición previa).
- **distribuir copias.**
- **mejorar el programa** y hacer públicas las mejoras (el código fuente es un requisito previo)

Un software **sigue siendo no libre aún si el código fuente es hecho público**, cuando se mantiene la **reserva de derechos sobre el uso, modificación o distribución.**

**Ventajas:**

- Es desarrollado por **grandes corporaciones** que emplean grandes recursos en su proceso y por lo tanto someten a sus productos a un periodo de **validación** mas largo antes de su implementación definitiva, lo que hace que sea **mas estable** desde su lanzamiento.

- Tiene un **uso mayoritario**, suele ser utilizado por organismos gubernamentales, universidades, empresas, aunque el software libre se está abriendo camino entre los ámbitos oficiales.

- **Soporte** que ofrece a sus clientes, ya que cuentan con una gran empresa detrás del software que, asegura su **estabilidad** y **perdurabilidad** en el tiempo.

**Inconvenientes:**

- Es un producto **cerrado** que **no** puede ser **modificado ni mejorado**, son los usuarios y las empresas las que deben adaptarse a él y no al contrario.
- Depende de un **empresa**. Como todas las empresas están sujetas a intereses económicos, por lo tanto pueden quebrar o ser absorbidas por empresas mayores, que **dejen de continuar con esa línea de desarrollo**.

*Ejemplo: Caso muy parecido es el que ocurrió con el conocido software de diseño vectorial Freehand, su empresa , Macromedia, fue absorbida por Adobe y el programa dejó de actualizarse hasta quedar en desuso en favor del software que ya comercializaba Adobe,  Adobe Illustrator.*

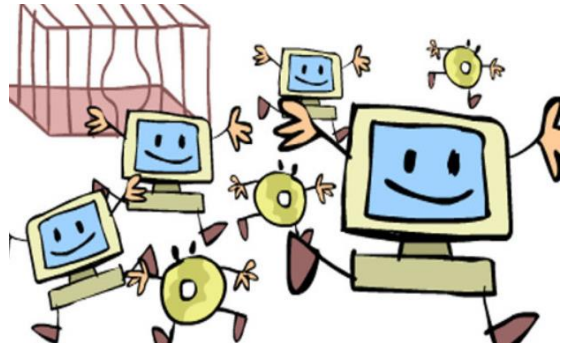
**Software libre**

Proporciona al usuario las siguientes **cuatro libertades siguientes**, **autoriza** para:

- **Utilizar** el programa, para **cualquier propósito**.
- **Estudiar** cómo funciona el programa y **adaptarlo** a tus necesidades, debe proporcionarse las **fuentes**, directa o indirectamente, pero siempre de **forma fácil y asequible**.
- **Distribuir** copias.
- **Mejorar** el programa y **hacer públicas las mejoras** a los demás.

Todo programa que no incorpore alguna de estas libertades se considera **no libre o semilibre**.
La mayor parte de las licencias de software libre **surgen de la FSF (Free Software Foundation)**.
El software libre suele estar disponible **gratuitamente** (o **precio de costo de la distribución**), sin embargo **no es obligatorio que sea así**, por lo tanto no hay que asociar software libre a "software gratuito" (denominado usualmente **freeware**), ya que, conservando su carácter de libre, **puede ser distribuido comercialmente**.

FREE SOFTWARE FOUNDATION



## Software de dominio público

Es aquél que **no está protegido con copyright y que no requiere de licencia**, pues sus **derechos** de **explotación son para toda la humanidad**.
Esto ocurre cuando el autor lo **dona a la humanidad** o si los **derechos de autor han expirado** (en un plazo contado desde la muerte del autor, generalmente 70 años).
En caso de que el autor condicione el uso de su software bajo una licencia, por muy débil que sea, ya no se consideraría software de dominio público.

## Software con copyleft

Es el **software libre** cuyos términos de distribución no permiten a los redistribuidores agregar ninguna restricción adicional cuando lo redistribuyen o modifican, o sea, la **versión modificada debe ser también libre**.


***Creative Commons*** *(CC): es una* ***organización*** *sin fines de lucro dedicada a* ***promover el acceso y el intercambio de cultura****. Desarrolla un conjunto de instrumentos jurídicos de carácter gratuito que facilitan usar y compartir tanto la creatividad como el conocimiento. No reemplazan a los derechos de autor, sino que se apoyan en estos para* ***permitir elegir los términos y condiciones de la licencia de una obra*** *de la manera que mejor satisfaga al titular de los derechos.* ***Una manera en que los autores pueden tomar el control de cómo quieren compartir su propiedad intelectual.***

**Otros tipos de software,** tales como:

- **Freeware**: Programa totalmente gratuito. Es posible que requiera que nos registremos, pero siempre de forma gratuita.
- **Shareware (Demo):** Se trata de una **versión reducida** del programa, con algunas funciones desactivadas para que podamos **probarlo** y decidir si lo vamos a comprar o no.


- **Shareware (Versión limitada por tiempo):** Se trata de una **versión totalmente funcional por un cierto número de días** (normalmente 30, puede variar según la compañía) tras la cual no lo podremos usar o se verá reducida su funcionalidad. Su objetivo es poder **probar** la aplicación y luego **decidir si la compramos o no**.

ZOPE


*Ejemplos de software libre.*

- *Sistemas Operativos: Debian GNU/Linux, Ubuntu, Guadalinex, Lliurex.*
- *Entornos de escritorio: GNOME, KDE.*
- *Aplicaciones de oficina: OpenOffice.*
- *Navegación web: FireFox, Konqueror.*
- *Aplicaciones para Internet: Apache, Zope.*



Actividad1 Software Propietario-Libre

# 2.- Sistemas Operativos.

El sistema operativo es un **conjunto de programas que se encarga de gestionar los recursos hardware/software del ordenador**, por lo que actúa como una **interfaz entre los programas de aplicación del usuario y el hardware puro**.

- Es un **elemento software** del sistema informático y por tanto pertenece a los elementos lógicos o **abstractos** del sistema informático
- Es un **conjunto de instrucciones**, **órdenes** y **datos** destinados a realizar un fin concreto.
- Pertenece a la categoría de **software base o básico**: para que un usuario pueda manejar un ordenador usará el SO como **intermediario,** el SO **ocultará** todo el funcionamiento interno y gestión del hardware (señales eléctricas, coordinación entre los componentes de la placa base, etc…).

Realmente el SO **no hace nada especial,** si iniciamos Windows10 y llegamos al escritorio, inicialmente si el usuario no ejecuta ninguna aplicación, entonces el **ordenador no hará nada.**

El SO tendrá **tres funciones** principalmente:

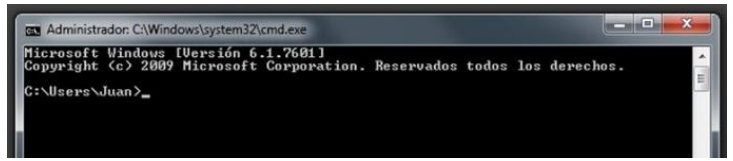
- **Controlar** la ejecución de las aplicaciones.
- **Explotar** y **Administrar los recursos hardware** del ordenador con el objeto de proporcionar un conjunto de servicios a los usuarios del sistema (procesador, memoria, periféricos, etc..): según administre estos recursos obtendremos sistemas informáticos diferentes y adaptados a diferentes entornos.

*Ejemplo un móvil, inicialmente, solo va a ser usado por una única persona física y por tanto no existe seguridad entre los documentos de diferentes usuarios que sí se necesita en los SO que se encuentran en una oficina, empresa*

- Proporcionar **interfaz de comunicación entre el usuario final y el hardware del ordenador**.

  Dos tipos de interfaces:

  - **modo texto** que se suele denominar intérprete de comandos, en donde el usuario debe **introducir los comandos** del sistema para realizar las operaciones oportunas, normalmente este tipo de interfaz está destinada a los **administradores** del sistema




  - un tipo de interfaz en **modo gráfico** en donde el uso del ratón predomina para **introducir las funciones** al sistema más **agradable** para los todos usuarios

## 2.1.- Objetivos de los Sistemas Operativos.

Los **principales objetivos** de los SO son:

- **Abstraer al usuario de la complejidad del hardware**: ordenador sea más fácil de utilizar.
- **Conveniente**: debe ser el más conveniente para cada ordenador

*Ejemplo: no podemos gestionar un ordenador de 256MB de memoria RAM con Windows10*

- **Eficiencia**: permite que los recursos del ordenador se utilicen de la forma más eficiente posible para obtener el mayor rendimiento del hardware.

*Ejemplo: optimizar los accesos a disco para acelerar las operaciones de entrada y salida*.

- **Permitir** la **ejecución de programas**: cuando un usuario quiere ejecutar un programa, el sistema operativo realiza todas las tareas necesarias para ello.

*Ejemplo: cargar las instrucciones y datos del programa en memoria, iniciar dispositivos de entrada/salida, asignar procesos a la CPU.*

- **Acceder a los dispositivos entrada/salida**: suministra una **interfaz homogénea** para los dispositivos de entrada/salida para que el **usuario pueda utilizarlos** de forma más sencilla.
- Proporcionar una **estructura y conjunto de operaciones para el Sistema de Archivos**.

- **Detección y respuesta ante errores (robustez):** debe prever todas las posibles situaciones críticas y resolverlas, si es que se producen. No debe quedar un sistema inestable.

- Controlar el **acceso al sistema y los recursos**: debe proteger las diferentes zonas de memoria vitales en donde se ubica el propio SO de los diferentes procesos del usuario, y en el caso de sistemas compartidos, proporcionando protección a los **recursos y los datos** frente a usuarios no autorizados.

- Capacidad de **adaptación/evolución**: construido de manera que pueda evolucionar a la vez que surgen actualizaciones hardware y software, de manera que permita el desarrollo, prueba o introducción efectiva de nuevas funciones sin interferir con el servicio.

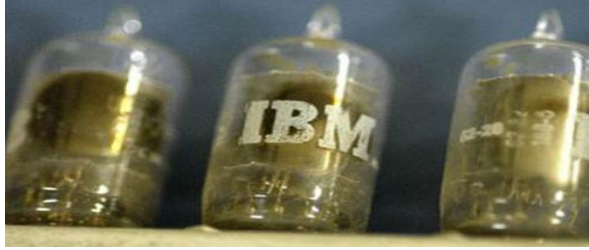
- Gestionar las **comunicaciones en red**: instalación, configuración y uso de las redes de ordenadores.

- Permitir a los usuarios **compartir recursos y datos**: relacionado con el anterior, da al SO el papel de **gestor** de los recursos de una red.

**Evolución histórica de los Sistemas Operativos**

El **hardware** y el **software** de los sistemas informáticos ha evolucionado de forma **paralela** y conjunta en las últimas décadas.
La evolución de los SO está estrechamente relacionada con los avances en la arquitectura de los ordenadores que se produjo de cada generación.

- **1ª Generación (1945-1955)**
  - Los primeros ordenadores estaban construidos con tubos de vacío.




  - **No** existían **Sistemas Operativos**, se programaba directamente sobre el **hardware**.
  - Los **programas** estaban hechos directamente en **código máquina** y el control de las funciones básicas se realiza mediante **paneles enchufables**.
  - Hacia finales de 1950 aparecen las **tarjetas perforadas** que sustituyen los paneles enchufables. Supusieron un gran paso: permitían **codificar instrucciones de un programa y los datos en una cartulina** con puntos que podía interpretar el ordenador.
  - La mayoría de los programas usaban **rutinas de E/S y un programa cargador** (automatizaba la carga de programas ejecutables en la máquina) esto constituía una **forma rudimentaria de sistema operativo**.

- **2ª Generación (1955-1965)**
  - Esta generación se caracteriza por la aparición de los **transistores** que permitieron la construcción de ordenadores **más pequeños y potentes**. Las computadoras se volvieron confiables para poder venderse a clientes con la esperanza de que continuaran funcionando el tiempo suficiente para realizar algún trabajo útil.


  - Las máquinas están en **cuartos de computadoras** especialmente **acondionados** con aire y con cuerpos de **operadores** profesionales para accionarlas
  - Para correr un trabajo, primero se escribía en papel, después se perforaba en tarjetas y se llevaba a la **pila de tarjetas del cuarto de introducción al sistema** donde se entregaba a un **operador**.
  - La programación se realizaba en **lenguaje ensamblador** y en **FORTRAN** sobre **tarjetas perforadas**.
  - **Procesamiento por lotes**, en el cual mientras el SO está ejecutando un **proceso**, éste último **dispone de todos los recursos hasta su finalización**. La preparación de los trabajos se realiza a través de un lenguaje de control de trabajos.

- El **SO residía en memoria** y tenía un programa de control que **interpretaba las tarjetas de control**. Dependiendo del contenido de la tarjeta de control el sistema operativo realizaba una acción determinada, es un antecedente de los modernos **intérpretes** de órdenes.
- **Procesamiento Fuera de línea** (Offline): como mejora del procesamiento por lotes surgió el procesamiento fuera de línea, las operaciones de **carga de datos y salida de resultados** de un proceso podían realizarse de **forma externa y sin afectar al tiempo que el procesador dedicaba a los procesos**. A esto ayudó la aparición de las **cintas** magnéticas y las **impresoras** de líneas.

- **3ª Generación (1965-1980)**
  - La aparición de los **circuitos integrados** (CI) supuso una mejora: un menor tamaño y relación precio/rendimiento respecto de las máquinas de generaciones anteriores.


  - La característica principal de esta generación fue **la multiprogramación** y **multiusuario**:
    - En los sistemas multiprogramados se cargan **varios programas en memoria simultáneamente** y se **alterna su ejecución,** lo que maximiza la utilización de la CPU.
    - En los sistemas multiusuario el tiempo del procesador se comparte entre programas de varios usuarios (multiusuario) pudiendo ser programas **interactivos**.

- **4ª Generación (1980-hasta hoy)**
  - Aparecen los **circuitos LSI (integrados a gran escala**, chips con millones de transistores en unos centímetros cuadrados), aparece el **microprocesador**.
  - Aparecen los **ordenadores personales**: **costo menor** y **mayores capacidades**.
  - Los SO evolucionan hacia sistemas **interactivos** con una **interfaz cada vez más amigable al usuario**. Aparición de numerosos **periféricos**.
  - **Desarrollo de redes de ordenadores** que ejecutan SO en **red** y SO **distribuidos**.
  - Aparecen **Sistemas de Gestión de BBDD**.
  - Actualmente, existen SO **integrados**, para una gran diversidad de **dispositivos electrónicos**, tales como, teléfonos móviles, dispositivos de comunicaciones e informática y electrodomésticos.

*Ejemplo: Android OS, un SO basado en Linux, diseñado en un principio para dispositivos móviles (como smartphones y tablets), pero también para su aplicación en netbooks y PCs.*

- software libre y de código abierto
- Android fue diseñado para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tablets; y también para relojes inteligentes, televisores y automóviles.
- Android es de código abierto, gratuito y no requiere pago de licencias.


Enlaces con información de cómo se crearon algunas de las empresas informáticas más importantes
https://histinf.blogs.upv.es/2011/12/17/historia-de-apple/
https://histinf.blogs.upv.es/2011/01/03/historia-de-microsoft/

## 2.2.- Tipos de Sistemas Operativos.

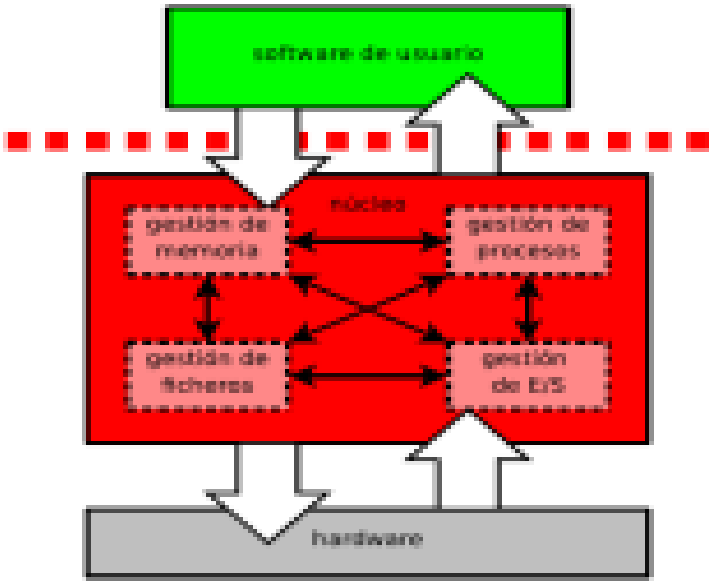
Clasificación de los SO en base a:

- **Estructura**
- **Servicios que suministran**
- **Forma**

| Tipos de sistemas operativos | | |
|---|---|---|
| Por estructura | Por sus servicios | Por su forma |
| Monolíticos | Monousuario | Sistema operativo en red |
| Jerárquicos | Multiusuario | Sistema operativo distribuido |
| Máquina Virtual | Monotarea | |
| Microkernel o Cliente-Servidor | Multitarea | |
| | Monoprocesador | |
| | Multiprocesador | |

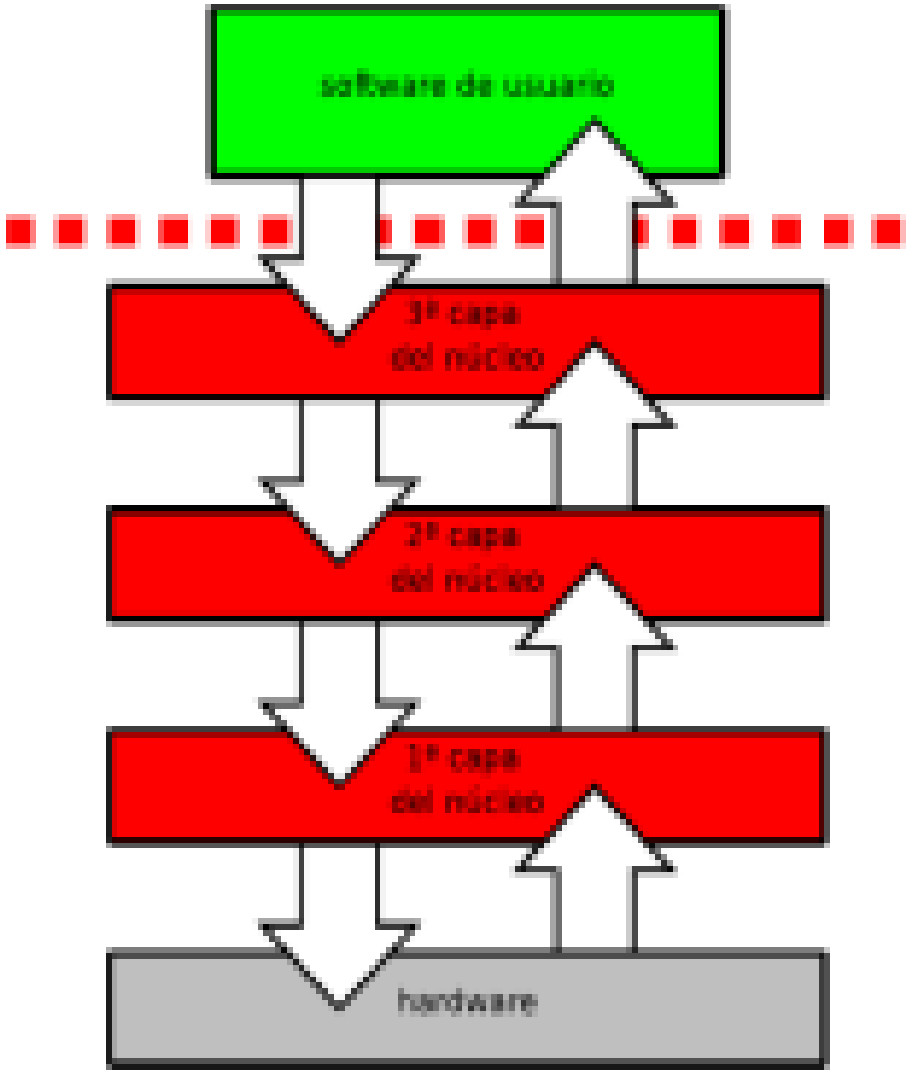
# Sistemas Operativos por su Estructura

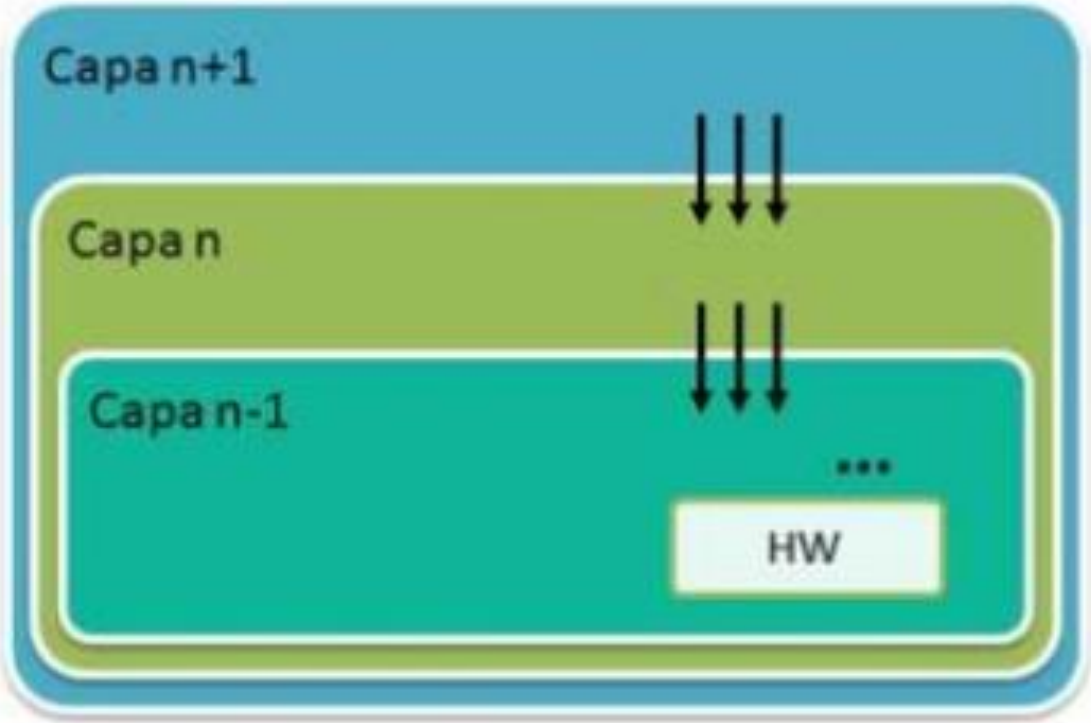
- **Monolíticos**: estructura de los primeros SO, **un único programa donde se integran todos los componentes. Desarrollado con rutinas entrelazadas** que podían llamarse entre sí. Son SO hechos a medida, pero difíciles de mantener y demasiado pesados.
  Si aparecen nuevas funcionalidades se agregan directamente al núcleo.



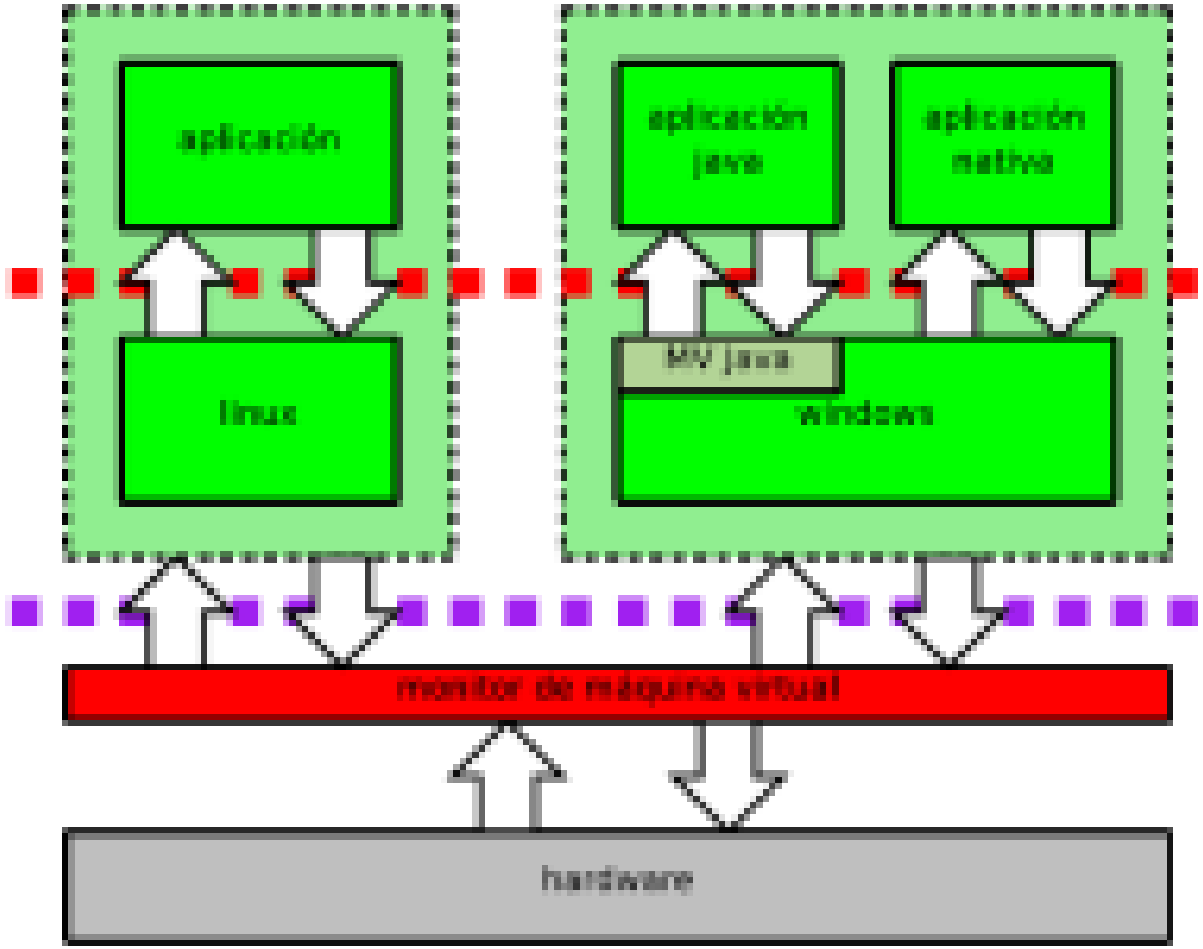

La construcción del programa final es a base de **módulos compilados separadamente** y que se unen a través de un compilador. **Carecen de protecciones y privilegios** al manejar recursos como memoria y disco duro.

- **Jerárquicos**: conforme las necesidades de los usuarios aumentaron, los SO fueron **creciendo en complejidad y funciones**. Se hizo necesaria una mayor *organización del software* del SO, **dividiéndose en partes más pequeñas**, diferenciadas por funciones o **capas** y con una **interfaz** clara para interoperar entre ellas y con los demás elementos.

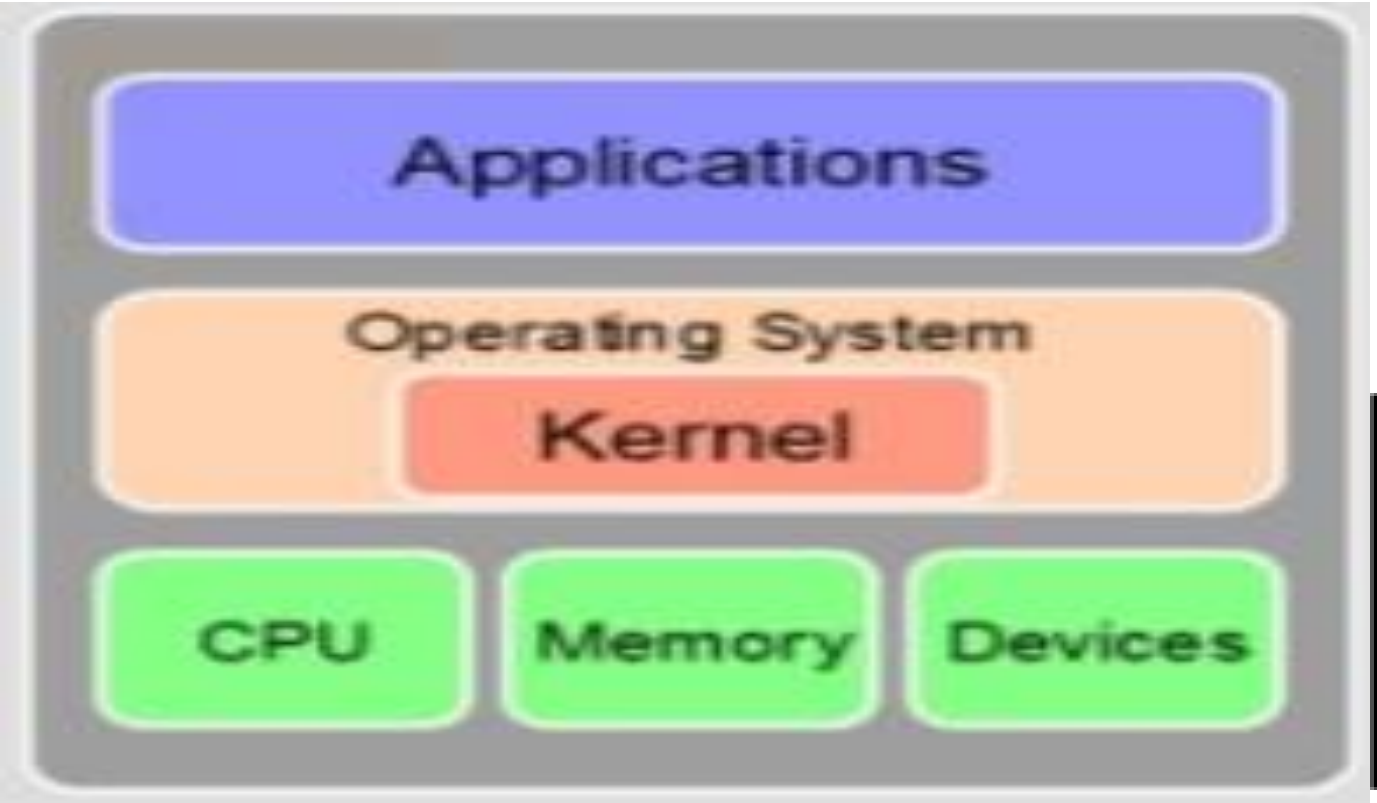


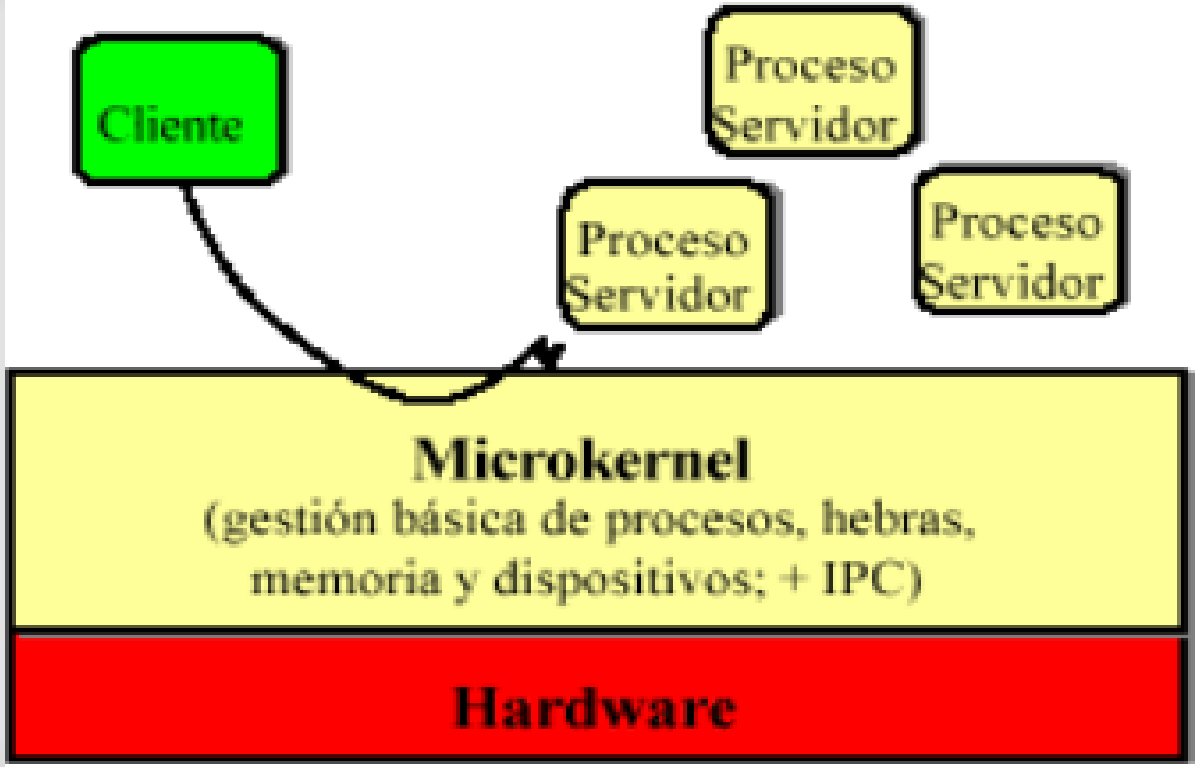



- **Máquina Virtual**: el objetivo es **integrar distintos SO** dando la **sensación de ser varias máquinas diferentes**, mostrando una **máquina que parece idéntica a la máquina real subyacente**. Son una **réplica** de la máquina real, en cada una de ellas **se pueda ejecutar un SO diferente**, que será el que ofrezca la máquina extendida al usuario.
  *Ejemplo de este tipo de sistemas: VMware y VirtualBox*



- **Cliente-Servidor:**
  - el más **reciente**, sirve para toda clase de aplicaciones y es de tipo general
  - **aísla** el núcleo (operaciones de entrada/salida, gestión de memoria, sistema de archivos) del resto del sistema informático.
  - sigue el esquema **cliente/petición** de aplicaciones por medio de **llamadas al sistema,** el **kernel** le facilita esas tareas con la gestión de procesos, memoria, entrada/salida, etc.
  - **distribuye las diferentes tareas en porciones de código modulares y sencillas**
  - incrementa tolerancia a fallos, seguridad y la portabilidad entre plataformas de hardware.



## Sistemas Operativos por sus Servicios

- **Monousuario**:
  - **soportan a un usuario a la vez**, sin importar el número de procesos o **tareas que el usuario pueda ejecutar en un mismo instante de tiempo**.
  - no existen restricciones a la hora de explotar el hardware, **no se definen perfiles o cuentas de usuario que puedan restringir las acciones a realizar** según se identifique en el dispositivo.
  - **tampoco existe protección** en los ficheros o documentos pues todos los documentos pertenecen al mismo usuario

  *Ejemplos: MS-DOS, Microsoft Windows 9x*

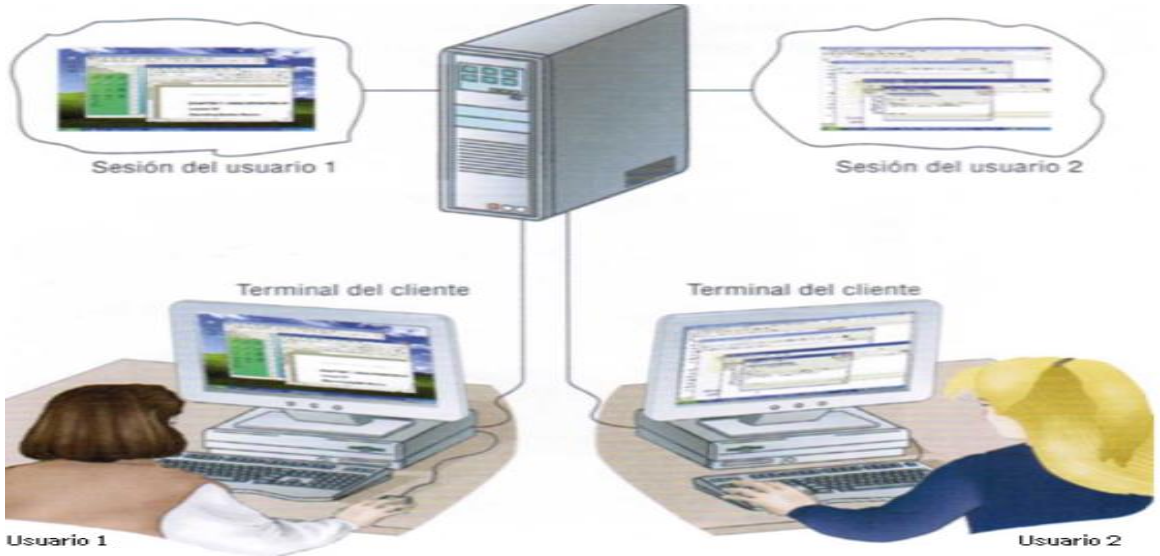
- **Multiusuario**:
  - dan **servicio a más de un usuario a la vez**, ya sea por medio de varios **terminales** conectadas al ordenador o por medio de **sesiones remotas** en una red de comunicaciones.
  - no importa el número de procesadores en la máquina ni el número de procesos que puede ejecutar cada usuario simultáneamente.

  *Ejemplos: UNIX, GNU/Linux, Microsoft Windows Server*

- se definen **cuentas de usuarios** (perfiles) que indican qué acciones se pueden realizar sobre el sistema (**administrador** cuenta que usará una persona para poder instalar software, configurar dispositivos, etc… y **usuario estándar** para los usuarios finales).

Podemos encontrar a su vez **dos tipos**:

- sistemas **multiusuarios locales:** son aquellos en donde el SO, las cuentas de usuarios y la explotación del sistema se realiza en el mismo equipo. Podemos definir diferentes cuentas de usuario e iniciar sesión con ellas, pero en el mismo equipo sin usar ningún servicio de red, con lo que cada cuenta de usuario solo se puede usar en el equipo en la que se define.
- sistemas **remotos** son aquellos en donde se **separa la localización de las cuentas de usuario y el equipo que se usa (Active Directory).** Los equipos **clientes** realizan el **login contra el servicio de directorio.**
  Favorece la movilidad dentro del espacio de explotación del sistema.



- **Monotarea**: sólo permiten **una tarea a la vez por usuario**. Se puede dar el caso de un sistema multiusuario y monotarea, en el cual se admiten varios usuarios simultáneamente pero cada uno de ellos puede ejecutar sólo una tarea en un instante dado.

*Ejemplos: MS-DOS*

- **Multitarea**: permite al usuario **realizar varias tareas al mismo tiempo**.

Dos o más tareas se ejecutan de forma **simultánea** cuando sus instrucciones se ejecutan en el mismo instante. Un procesador será capaz de ejecutar tantas tareas de forma simultánea como **núcleos** tenga y eso actualmente es un número muy inferior al número de procesos que existen en un sistema.

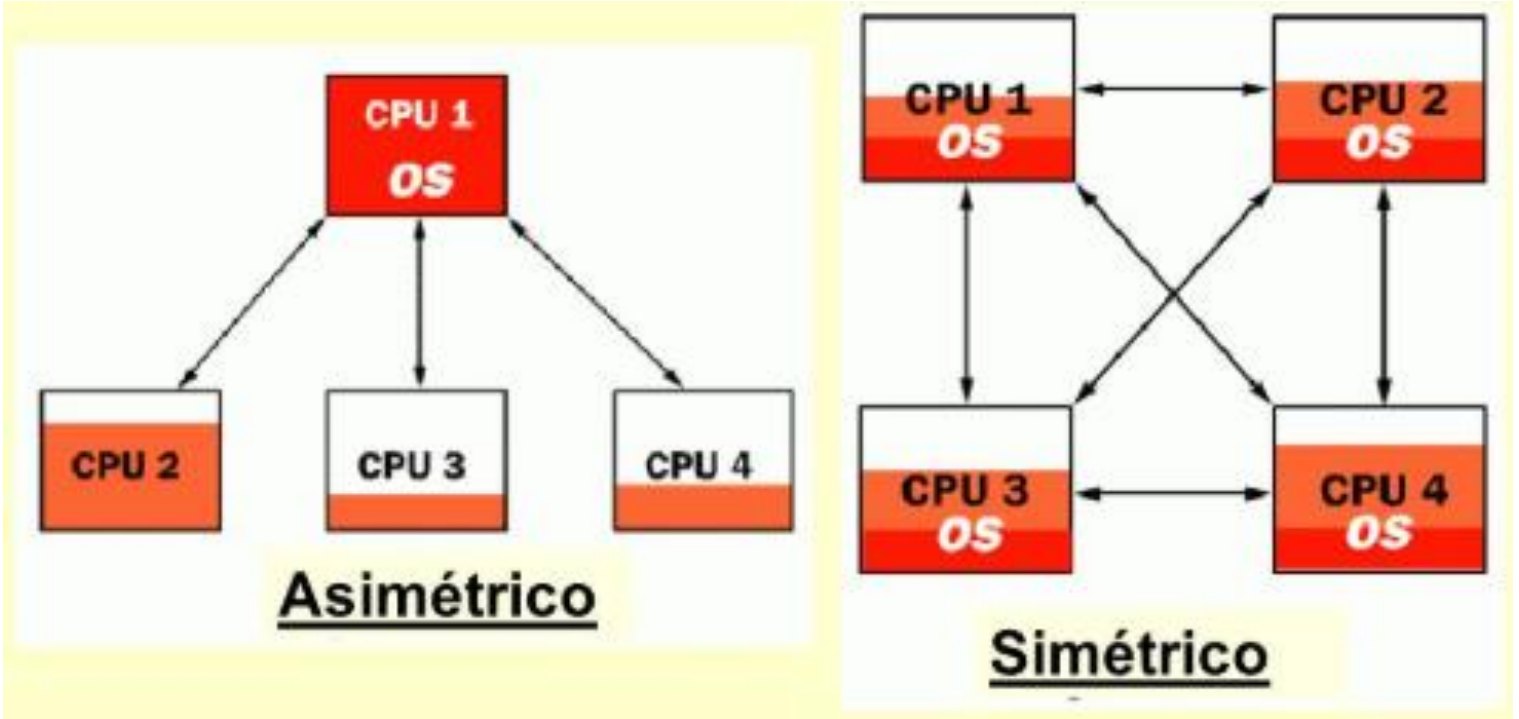
***Multitarea:*** *posibilidad de los SO de* ***gestionar la memoria y los procesos para que pueda albergar varios procesos que se van a ir ejecutando de forma solapada*** *unos con otros de tal forma que se va a* ***aparentar la ejecución simultánea de dichas tareas.***

*Ejemplos: Linux, Microsoft Windows 2000/7/10*

- **Monoprocesador**: capaz de **manejar sólo un procesador**, de manera que si el ordenador tuviese más de uno le sería inútil.

*Ejemplos: MS-DOS*

- **Multiprocesador**: se refiere al **número de procesadores del sistema**, éste es más de uno y el SO es **capaz de utilizarlos todos para distribuir su carga de trabajo**.
  Estos sistemas trabajan de dos formas:
  - **Simétricamente**: procesos son enviados **indistintamente** a cualquiera de los procesadores.
  - **Asimétricamente**: uno de los procesadores actúa como **maestro** o servidor y distribuye la carga de procesos a los demás.



## Sistemas Operativos por su Forma

- SO en **red**: tienen la capacidad de **interactuar con los SO de otras máquinas** a través de la red, con el objeto de **intercambiar información**, transferir archivos, etc.

El **usuario debe conocer la ubicación** de los recursos en red a los que desee acceder.

*Ejemplos: SO modernos Windows Server/10, Linux.*

- SO **distribuidos**: abarcan los servicios de red, las funciones se distribuyen entre diferentes ordenadores, logrando **integrar recursos** (impresoras, unidades de respaldo, memoria, procesos, etc.) en **una sola máquina virtual que es a la que el usuario accede de forma transparente**.

El usuario **no necesita saber la ubicación** de los recursos, sino que los referencia por su nombre y los utiliza como si fueran locales a su lugar de trabajo habitual.

## 2.3.- Componentes de un Sistema Operativo.

Un SO básicamente está formado por:

- El **núcleo**
- Los **servicios**
- El **intérprete** de órdenes o **shell**

### 2.3.1.- **Núcleo** (Kernel)

Es la parte que **interacciona** directamente con el **hardware** del equipo, permitiendo a las aplicaciones que accedan a estos de forma segura.

Es la parte **principal** del código de un SO y se encarga de la **gestión/petición** de los **recursos** hardware y **controlar/administrar los servicios**.
*Ejemplo: Si una aplicación necesita el uso de memoria, realizará una llamada al sistema para alertar al gestor de memoria.*

El núcleo normalmente representa sólo **una pequeña parte** de todo lo que es el SO, pero es una de las **partes que más se utiliza**. Reside por lo general en la **memoria principal**, mientras que otras partes del SO son **cargadas** en la memoria principal solo cuando se necesitan.

## 2.3.2.- Servicios de los Sistemas Operativos.

Cuando un usuario **ejecuta una aplicación**, esta necesitará la utilización de los componentes hardware, es decir, la **aplicación solicitará el servicio** que necesite.

Estos servicios para comunicarse con las aplicaciones es la **interfaz de llamadas al sistema** (API)

Los principales **recursos** que administra el SO son:


- **Gestor de Procesos:** asignar que proceso debe usar en que momento el procesador.
- **Gestor Memoria Principal:** partes libres/ocupadas y asignación/liberación de procesos.
- Gestor **Entrada/Salida:** dispositivos periféricos.
- **Sistema de Archivos:** gestión del almacenamiento secundario, gestión de ficheros/directorios.

***Los SO según su estructura se dividen en:***

- *Monolíticos, Jerárquicos, Monotarea y Multitarea.*
- *Monolíticos, Jerárquicos, Microkernel y en red.*
- *Monolíticos, Jerárquicos, Microkernel y Máquina virtual.*
- *Monolíticos, Jerárquicos, Máquina virtual, Microkernel y Distribuidos.*

***Los servicios principales que presta un sistema operativo son:***

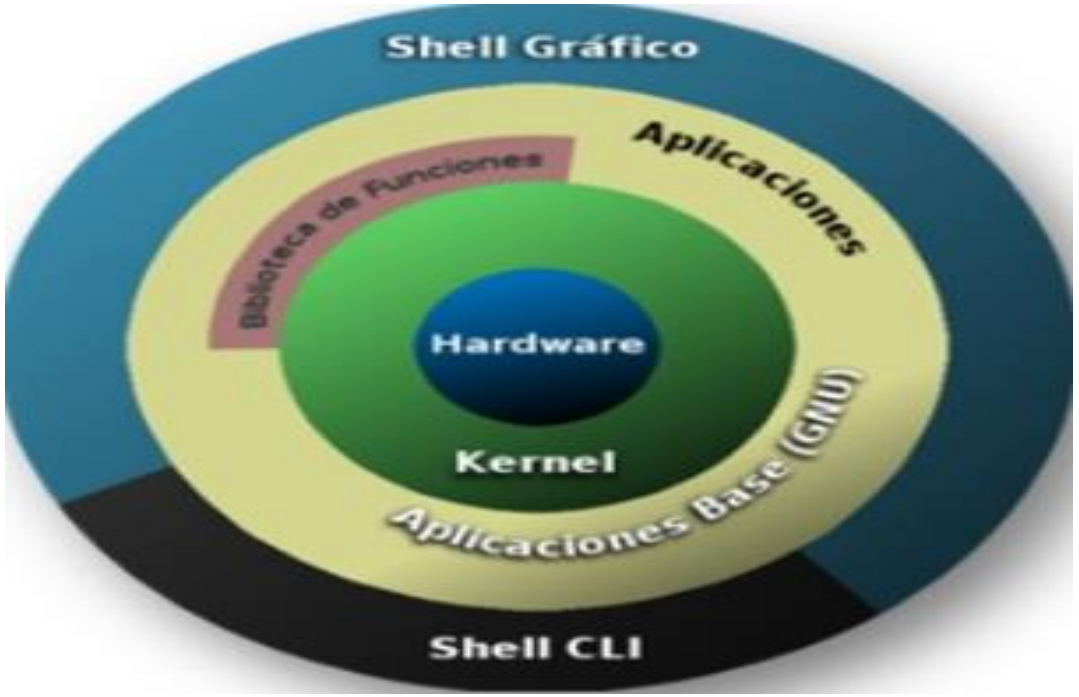
- *Gestión del Procesador, gestión de Memoria y de E/S.*
- *Gestión del Procesador, gestión de Memoria, de E/S y del Sistema de Archivos*
- *Gestión del Procesador y gestión de Memoria.*
- *Gestión de Memoria, de E/S y del Sistema de Archivos.*

## 2.3.3.- El intérprete de comandos/órdenes o Shell.

Proporciona al usuario una **interfaz** por la que poder hablar con el SO.

El usuario dispondrá de una serie de **comandos** que el Shell interpretará proporcionando información o **realizando alguna operación concreta**.

Según el SO el Shell variará y las órdenes serán diferentes.




Actividad2 Windows vs Linux
Actividad3 Características Sistemas Operativo

# 3.- Gestión de procesos.

Un **proceso** es la unidad de ejecución de instrucciones y órdenes procedentes del software.

Ocupan **memoria interna y CPU.**
Tienen un **estado** (variables con valores concretos).

No confundir programa con proceso:

- **programa** es un conjunto **completo** de instrucciones que sirven para solucionar un problema, es completo ya que hay que **expresar todos los posibles casos que puedan ocurrir**.

*Ejemplo podemos hacer un programa que dado un número entre 1 y 10 nos indique la nota escrita (notable, bien, sobresaliente, etc…)*

- **proceso** es la ejecución del programa, sólo una opción será la que se ejecute para una instancia concreta.

*Ejemplo: en el programa anterior indica la nota para un número concreto.*

El SO sigue la pista de en qué estado se encuentran los procesos y decide en que estado quedan:

- **gestiona los cambios de estado de los procesos.**

Concepto de **hilo de ejecución(thread)**.

Un procesos tienen al menos un hilo de ejecución que se encarga de ejecutar el programa.

Debido a la evolución del hardware, los **desarrolladores** de software para sacar el máximo **rendimiento** al hardware **diseñan** y crean aplicaciones que se pueden **dividir en subprocesos** internos que será ejecutados por hilos independientes dentro del proceso principal.

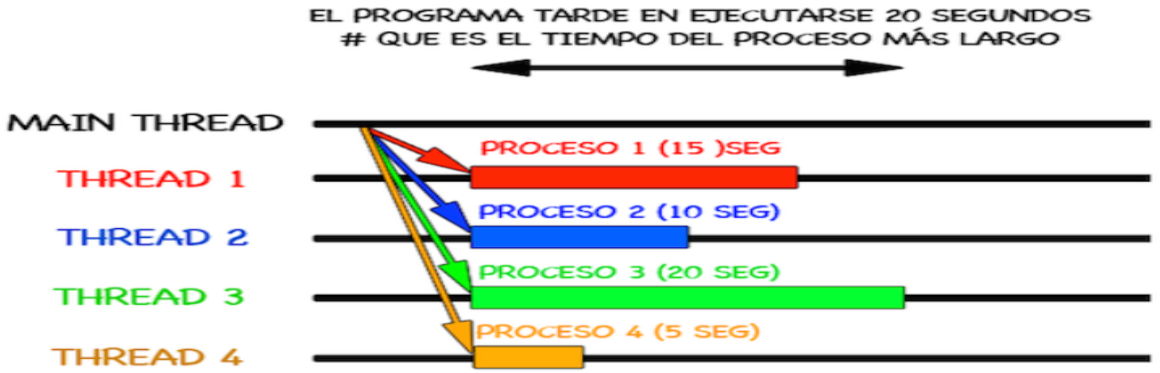
Esto hecho es **transparente al SO**, es el **programador** el encargado de crearlo, destruirlo, asignarle las acciones a realizar, coordinarlo con otros hilos dentro del proceso, etc…

Un hilo es, pues, una **tarea que puede ser ejecutada en paralelo con otra tarea**. Por lo que los hilos de ejecución permiten a un programa **realizar varias tareas a la vez**.

***Ejemplo**: en los procesadores de texto mientras escribes, el procesador te indica las faltas de ortografía y gramática, hay dos hilos de ejecución (dos subprocesos dentro del proceso principal Word). En los editores de texto de los años 90, la verificación de ortografía/gramática el usuario dejaba de escribir y elegía la acción dentro del menú, mientras no se podía escribir.*

***Ejemplo**: puedes tener varias pestañas con diferentes páginas webs (varias conexiones simultáneas, cada una de ellas un hilo de ejecución dentro de la aplicación del navegador)*

***Ejemplo**: chats en los que podemos tener varias conversaciones simultáneas, mientras se escribe se reciben notificaciones, etc.*

**Tipos de Procesos**

Según la forma en la que los **procesos interactúan con el usuario final**, podemos encontrar **tres:**

- **aplicaciones:**
  - procesos muy **interactivos** con el usuario, necesitan mucha **comunicación con el usuario. final** para realizar su cometido, **poca necesidad de procesamiento**, bajo % de uso de CPU.

  *Ejemplo: procesador de texto no usa la CPU ya que la introducción de texto en sí misma no necesita grandes cantidades de procesamiento.*
  - procesos **con mucha necesidad de procesamiento.**

  *Ejemplo: editores gráficos o de videos son tareas o aplicaciones que sí necesitan una gran capacidad de CPU.*

- **servicios:**
  - poco o **nada interactivos** con el usuario.
  - administrador del sistema configura el servicio y éste se **ejecuta de forma que el usuario final no nota la ejecución** de dicho proceso.
  - suelen ser **procesos del SO** o de fabricantes de hardware.

  *Ejemplos: actualizaciones automáticas (Windows Update), servicio de conexión inalámbrica.*

- **aplicaciones en segundo plano:**
  - procesos **mixtos** entre aplicaciones y servicios.
  - pueden **ejecutarse sin** prácticamente intervención por parte del usuario (2º plano).
  - tienen una **interfaz** completa de **comunicación con el usuario**.

  *Ejemplo: antivirus, gestores de descarga, suelen ser aplicaciones que se encuentran en el área de notificación, el usuario configura el proceso y lo deja latente y mientras que realiza el trabajo el usuario no interactúa con él.*


Existe una **segunda clasificación** de los procesos según su **forma en la que se ejecutan**:

- **intensivos en E/S:** concuerdan con las características de las aplicaciones, donde la interacción con el usuario es grande y el usuario debe introducir información frecuentemente.
- **intensivos de CPU**: procesos con gran capacidad de cálculo (renderizado, aplicación de filtros a imágenes, generación de informes, etc…).

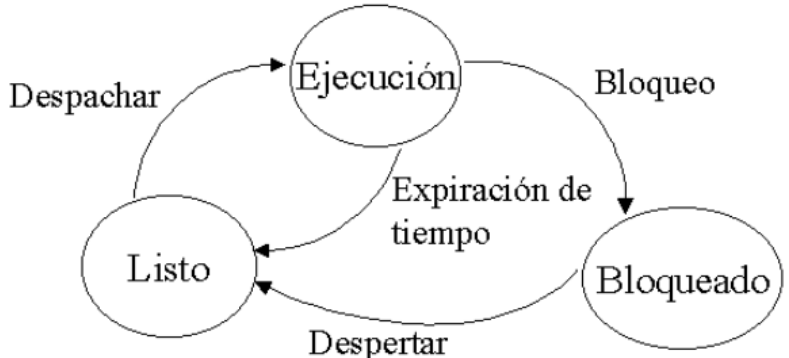
Los procesos **alternan ráfagas de uso de unidad central de proceso con ráfagas de acceso a los dispositivos** y siempre empiezan y terminan con una ráfaga de CPU.
En este concepto es en el que se basa la **Multitarea.**

**Estados de los Procesos.**

La **multitarea** es posible gracias a que los procesos alternan ciclos de ejecución con ciclos de acceso a los dispositivos.

Los procesos pueden tener **diferentes estados**.
**Simplificando**, en un instante dado, pueden estar en uno de los **tres estados siguientes**:

- **Listo/Espera/Preparado**: preparados para **poder ser elegidos** para hacer **uso** de la **CPU**.
- **Ejecución**: indica que el proceso está actualmente **usando la CPU** y está **ejecutando sus instrucciones**. Sólo puede haber un único proceso en este estado, salvo que existan varias unidades de proceso (núcleos).
- **Bloqueado**: procesos que han terminado su ciclo de CPU y están **a la espera obtener alguna información o dato externo** (ciclo de acceso a los dispositivos E/S) para poder continuar su ejecución.



**Ciclo de vida de un proceso,** paso por los **diferentes estados básicos:**

- Un proceso cuando se **carga en memoria** está **preparado** para ejecutarse y realizar su primera ráfaga de CPU, pero tendrá que esperar a que le toque su turno ya que posiblemente otro proceso esté usando la CPU.
- Cuando le **toca su turno** (lo decide el planificador) entonces pasa a **Ejecución**, en este estado **ejecuta instrucciones** y pueden pasar **tres** cosas:
  - termine y pase a **Finalizado.**
  - requiera un ciclo de acceso a los dispositivos con lo que pasaría a **Bloqueado.**
  - el planificador decida que otro proceso tiene que ejecutarse, interrumpe al proceso en curso, con lo que el proceso interrumpido pasa a **Preparado.**

- Por último cuando un proceso que está en estado bloqueado **termina su ciclo de acceso a los dispositivos pasa a estado Preparado** y tendrá que esperar a que le vuelva a tocar el turno.

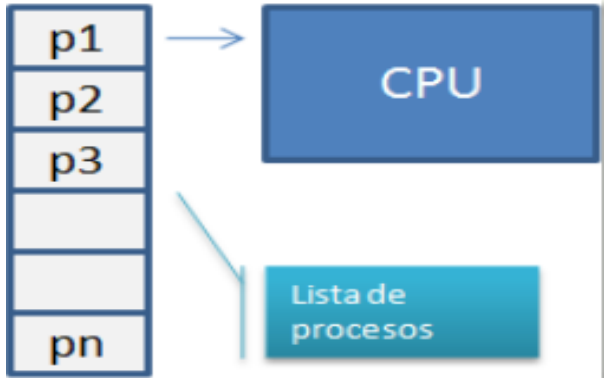
## 3.1.- Planificación del procesador.

## El Bloque de Control de Proceso (BCP)

Los procesos en estado **listo** pueden pasar a **ejecución** si el **planificador del SO lo selecciona.**
El planificador decide **cuánto tiempo de ejecución** se le asigna al proceso **y en qué momento**.

Si el sistema es monousuario y **monotarea** no habrá que decidir nada.
En los sistemas **multitarea** esta decisión es **fundamental para el buen funcionamiento del sistema**, ya que determinará la correcta ejecución de los programas que se estén ejecutando.

Los SO necesitan **mantener información sobre los procesos**, por cada proceso el SO mantiene lo que se denomina el bloque de control del proceso, estructura donde se guarda la **información necesaria para cada proceso**. **Todos los BCP** de todos los procesos se mantienen en memoria en la **Tabla de Control de Procesos**.



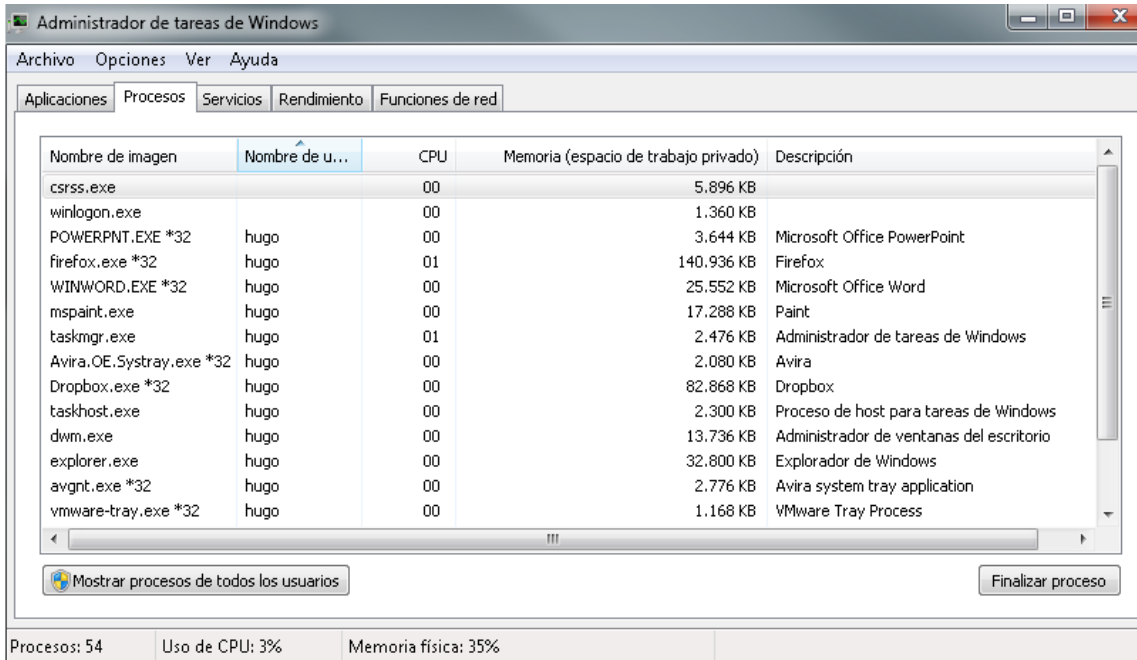

¿**Qué información se guarda** en el BCP? Depende del SO, pero en general, podemos encontrar:

- **Identificación** del proceso: bien con el **fichero** ejecutable del proceso o con un **número** de identificación del proceso (**PID**).
- Identificación del **proceso padre**.
- **Propietario** del proceso: cuenta de **usuario y grupo** que lanzó la ejecución del proceso y que indicará qué acciones o restricciones tiene ese proceso en el sistema multiusuario.
- **Estado** en el que se encuentra el proceso.
- Contenido de los **registros** internos, contador de programa, etc. El entorno **volátil** del proceso.
- Información de **control** de proceso necesaria para el planificador.

*Ejemplo: prioridad de un proceso.*

- **Segmentos de memoria** asignados (zonas de memoria en la que se aloja o cantidad de memoria que está usando el proceso).
- **Recursos** asignados.
- **Porcentaje** de uso de CPU.

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.3  0.0   2288   752 ?        Ss   21:25   0:02 init [2]
root         2  0.0  0.0      0     0 ?        S    21:25   0:00 [kthreadd]
root         3  0.5  0.0      0     0 ?        S    21:25   0:04 [ksoftirqd/0]
root         6  0.0  0.0      0     0 ?        S    21:25   0:00 [watchdog/0]
root         7  0.0  0.0      0     0 ?        S<   21:25   0:00 [cpuset]
```



En los entornos multitarea, lo que se pretende es **mantener ocupada la CPU la mayor cantidad de tiempo posible** y gracias a las características de los procesos a la hora de ejecutarse, lo que se hace es **solapar los procesos para que al final se obtenga un buen rendimiento**.

El **planificador** de procesos va a ser la parte del SO que se va a encargar de elegir qué proceso **preparado** pasa a ser **ejecutado**.
Según las **políticas** de decisión  por parte del planificador obtendremos valores de rendimiento diferentes. Tenemos que **definir parámetros** que **miden** estos **aspectos** a tener en cuenta en la ejecución de procesos en sistemas multitarea.

Una **estrategia** de planificación debe buscar que:

- los **procesos obtengan sus turnos de ejecución** de forma apropiada (uso de CPU)
- un buen **rendimiento**
- **minimización de la sobrecarga** (overhead) del planificador mismo.

Los **parámetros** a tener en cuenta son:

- **Equidad**: todos los procesos deben ser atendidos, es decir, **en algún momento obtienen su turno de ejecución o intervalos** de tiempo de ejecución hasta su terminación con éxito. Esto sirve para **evitar el aplazamiento indefinido**, los procesos deben **terminar** en un **plazo finito de tiempo**. El usuario no debe percibir que su programa se ha parado o "colgado".

- **Eficacia y Rendimiento**: el procesador debe estar **ocupado el mayor tiempo posible,** ha de maximizar el número de tareas que se procesan por unidad de tiempo, debe **finalizar el mayor número de procesos por unidad tiempo**.

- **Tiempo de espera**: es el **tiempo que pasa un proceso en espera desde que se lanza hasta que termina su ejecución**. En la cola de **Preparados**.

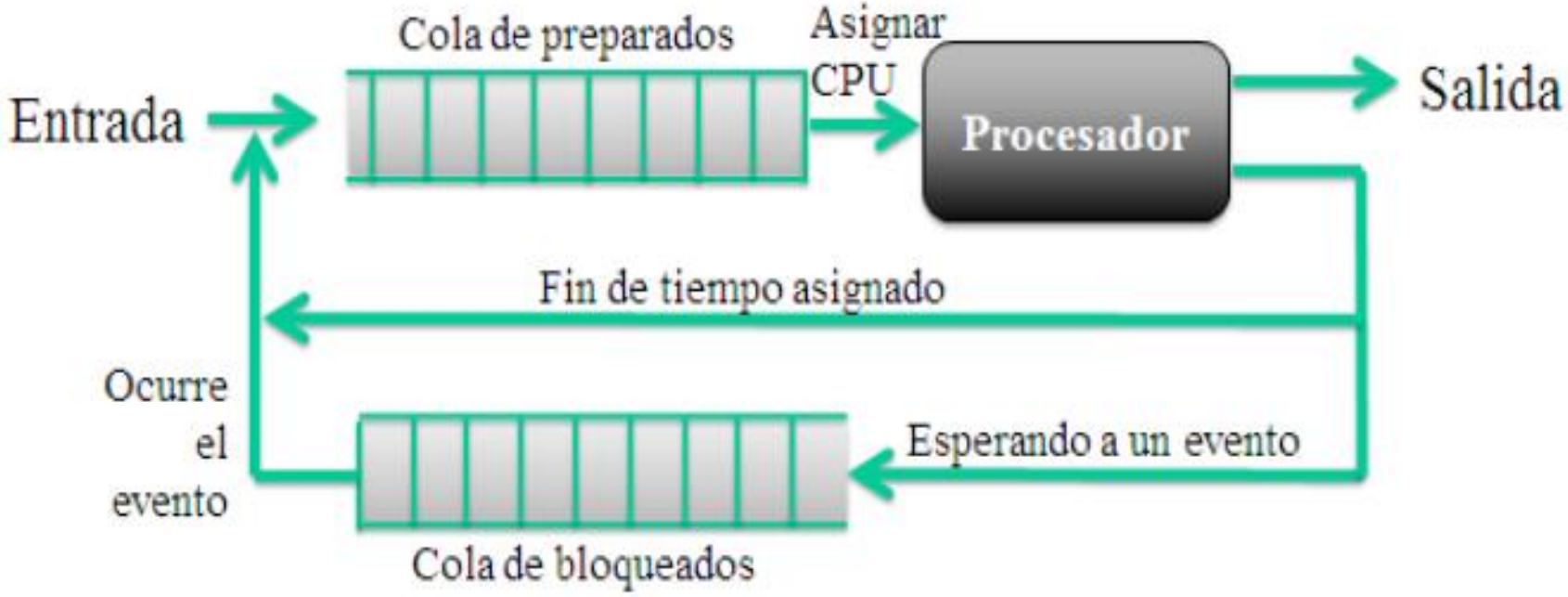
- **Tiempo de respuesta o retorno**: es el tiempo empleado en **dar respuesta a las solicitudes del usuario,** debe ser el menor posible. Es el tiempo que transcurre desde que se lanza hasta que finaliza por completo y muestra los resultados. El usuario no debe percibir tiempos de espera demasiado largos.

La **carga de trabajo** de un sistema informático puede **variar considerablemente**, esto dependerá de las **características** de los procesos.

Podemos encontrar Procesos:

- que hacen un **uso intensivo de la CPU**.
- que realizan una **gran cantidad de operaciones de Entrada/Salida**.
- por **lotes**, **interactivos**, en **tiempo real**.
- de **menor o mayor duración**.

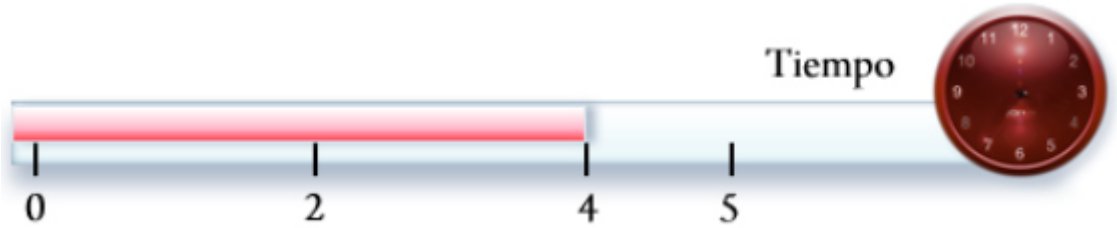
En función de cómo sean la mayoría de los procesos habrá **algoritmos de planificación que den un mejor o peor rendimiento** al sistema.



## 3.2.- Planificación apropiativa y no apropiativa.

- **No Apropiativa** (no preemptive): cuando a un proceso le toca su turno de ejecución, ya **no puede ser suspendido**, es decir, **no se le puede arrebatar el uso de la CPU** hasta que el proceso no lo determine (fin de ejecución u operación de E/S).
  **Problemas**:
  - si el proceso contiene **ciclos infinitos**, el resto de los procesos pueden quedar aplazados indefinidamente.
  - los **procesos largos** penalizarán a los cortos si entran en primer lugar.

- **Apropiativa** (preemptive): el SO **puede arrebatar el uso de la CPU a un proceso que esté ejecutándose**. Existe un **reloj** que lanza **interrupciones** periódicas en las cuales el **planificador toma el control y decide si el mismo proceso seguirá ejecutándose o se le da su turno a otro** proceso (mayor prioridad), con lo que un proceso en **Ejecución** puede pasar a **Preparado** sin haber terminado su ciclo de uso de la CPU o sin haber hecho solicitud de operación de E/S.



En ambos enfoques de planificación se pueden establecer distintos **algoritmos de planificación de ejecución de procesos**.
Un algoritmo de planificación es un **procedimiento que en base a cierta información de los procesos determinará qué proceso debe ejecutarse o no**.

Los **parámetros** estudiados anteriormente nos indicarán cómo ha mejorado o no un sistema con la elección de diferentes criterios.

Vamos a explicar algunos de los algoritmos de planificación, aunque realmente los SO actuales no usan ninguno al 100% sino que **mezclan o realizan pequeñas variaciones**.

Algunos de los algoritmos para **decidir el orden de ejecución** son:

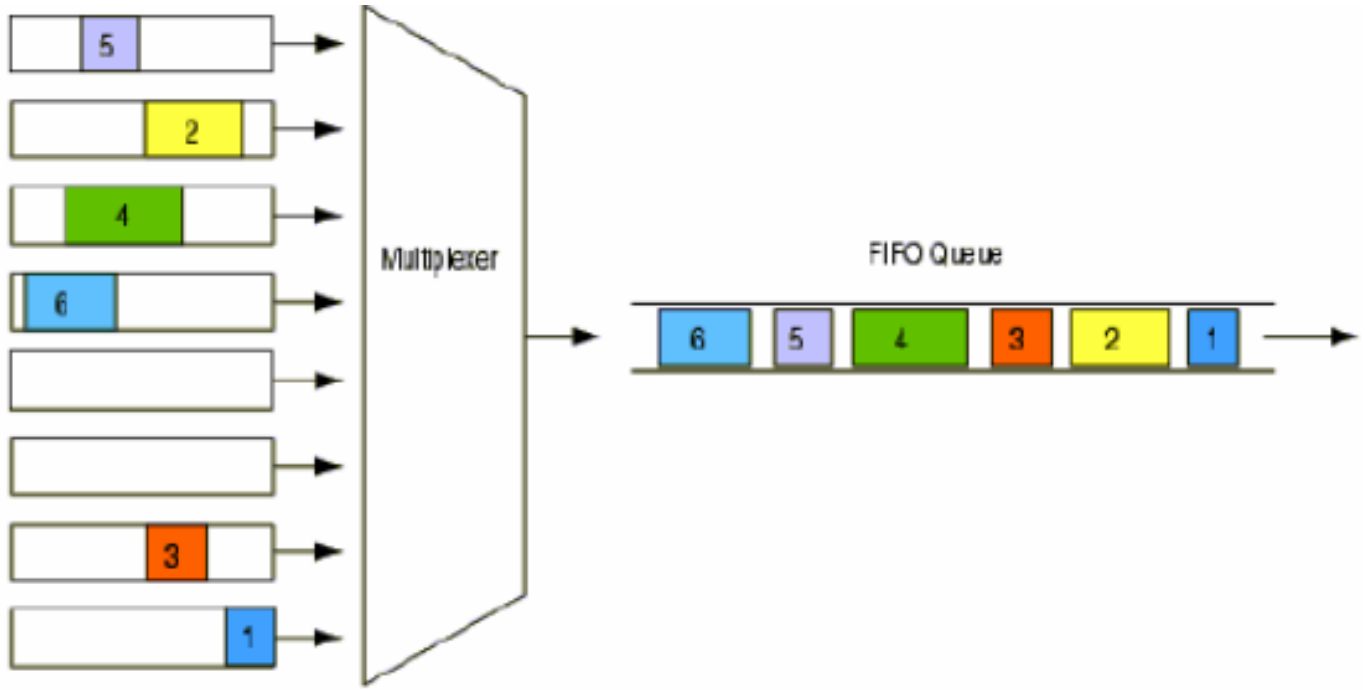
- Round Robin.
- Prioridad.
- Trabajo más corto primero.
- Primero en llegar, primero en ejecutarse.
- Tiempo ejecución restante más corto.

- **Algoritmo de planificación FIFO** (First Input First Output)
  - orden de ejecución de los procesos es el mismo que el **orden en el que se van creando**
  - planificación **no apropiativa,** ráfagas de CPU de un proceso no pueden ser interrumpidas por otros procesos

*Ejemplo: cola a la hora de pagar en un supermercado.*

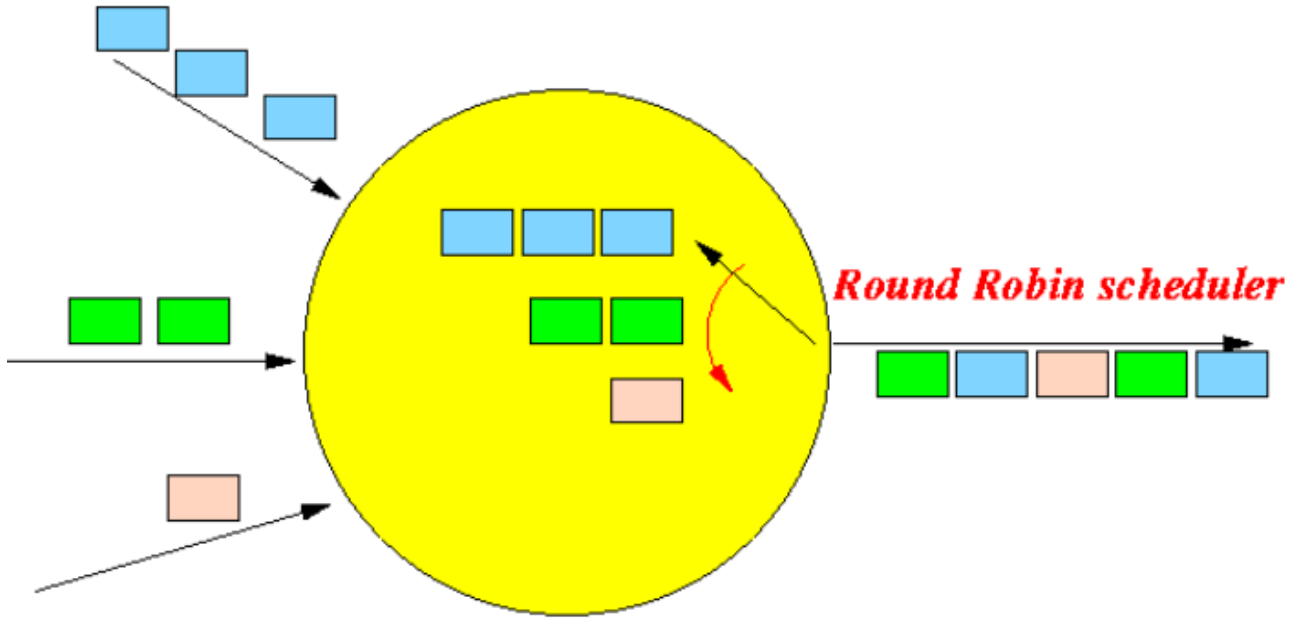
**Ventaja**: **fácil** implantación

**Inconveniente**: un **proceso muy largo monopolice la CPU** durante mucho tiempo generando tiempos de espera mayores de los que serían deseables.

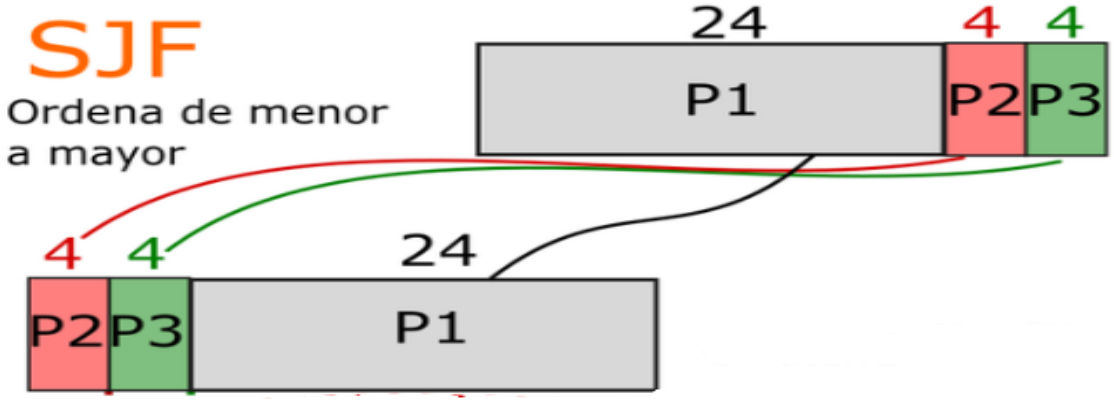



Actividad. Ejercicio Gestión Procesos

- **Algoritmo de planificación Round Robin** (RR)
  - política **apropiativa**
  - basado en la asignación de un tiempo máximo de ejecución (uso de CPU): el **quantum**
  - **tiempo total de ejecución** de un proceso se **divide en porciones** en base al quantum
  - se van **alternando** porciones de quantum de diferentes procesos que se ejecutan de forma **secuencial** según el orden de llegada. Cuando un proceso termina su quantum libera la unidad central de proceso y se **colocará al final de la cola** hasta que le vuelva a tocar su ejecución de otro quantum.



- **Algoritmo de planificación SJF** (Short Job First: Primero el trabajo más corto)
  - tiene en cuenta **orden de llegada** y **tiempo que tarda en ejecutar su ráfaga de CPU**, se **ejecutará primero aquel proceso que tenga una ráfaga de CPU más corta**.
  - política **no apropiativa**: una vez elegido un proceso que se ejecuta, éste terminará su ráfaga sin ser interrumpido.
  - **Inconveniente**: dificultad en saber el tiempo total de CPU que necesita.




- **Algoritmo de planificación SRTF** (Short Remaining Time First: Primero el trabajo de Tiempo Restante más Corto)
  - política **apropiativa**, lo que implica que mientras que un proceso está en su ráfaga de CPU, podrá ser interrumpido por otro proceso cuya ráfaga de CPU sea más corta de lo que le queda al primer proceso en terminar.

**Inconveniente**: **retrasan la ejecución de los procesos con ráfagas grandes de CPU,** incluso cabría la posibilidad de no ejecución de este tipo de procesos.

- **Algoritmo de prioridad**
  - Establece a cada proceso una **prioridad** (número) y en base a ese número se ejecutará el proceso con mayor prioridad.
  - Dos versiones:
    - **apropiativa**
    - **no apropiativa**

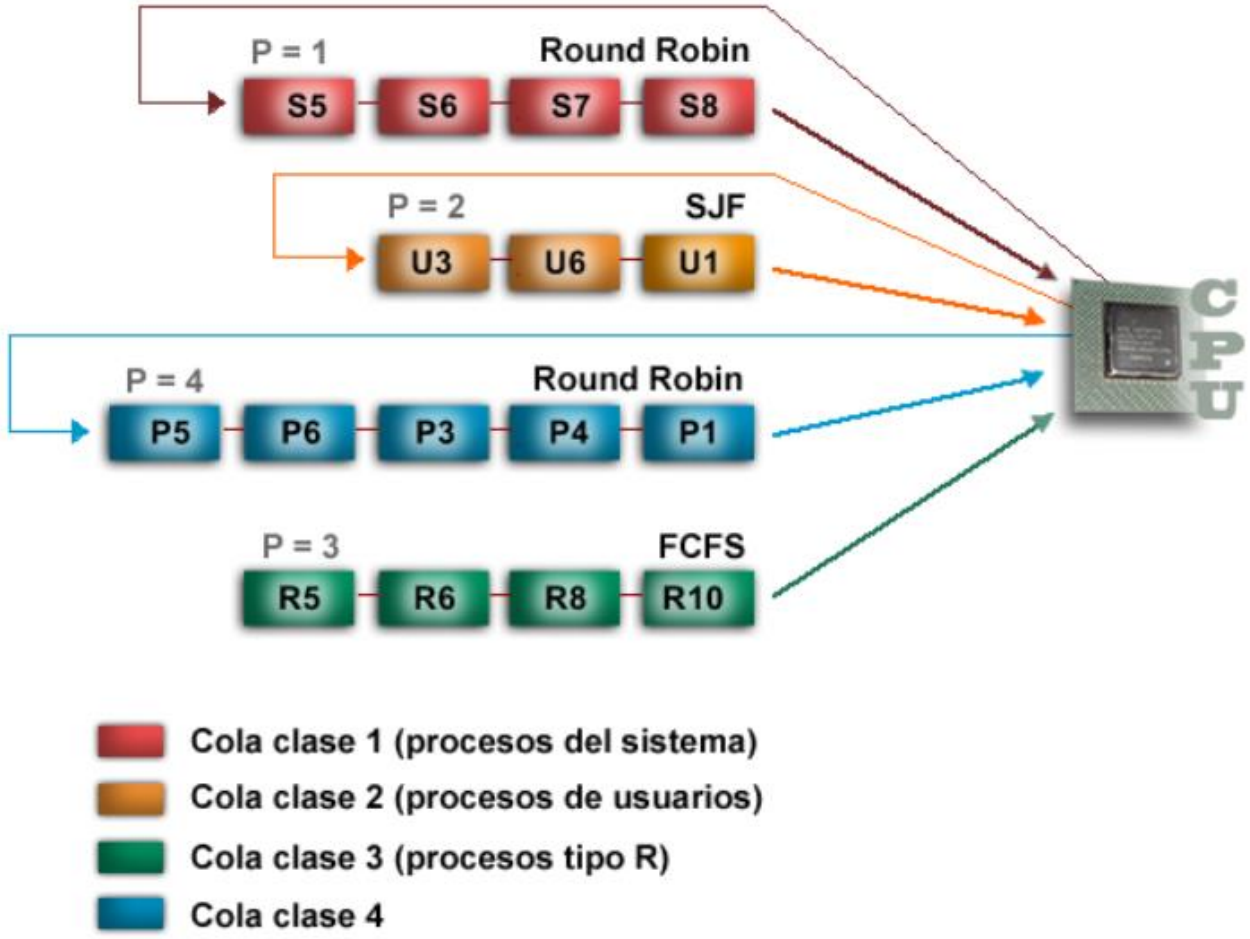
| Proceso | Prioridad | Tiempo de ejecución (te) |
|---|---|---|
| P1 | 3 | 10 |
| P2 | 1 | 1 |
| P3 | 3 | 2 |
| P4 | 2 | 5 |

  - **Problema**: posibilidad de que un proceso no se ejecute nunca o tarde mucho.

*Ejemplo: supongamos un proceso con prioridad baja y otro con prioridad alta, se va a ejecutar el proceso de prioridad alta. Mientras se ejecuta el proceso de prioridad alta se inicia un proceso de prioridad media pero superior al proceso de prioridad baja, esto implica que el proceso de prioridad baja se ejecutará después del de prioridad media. Mientras se ejecuta el de prioridad media, se inicia otro de prioridad alta y así sucesivamente, el proceso de prioridad baja tendrá un alto tiempo de espera en caso de ejecutarse ya que su ejecución se retrasa mucho en el tiempo.*

  - **Solución**: establecer asignación de prioridades **dinámicas** de tal forma que aquellos de prioridad baja, **vayan incrementando su prioridad conforme aumenta su tiempo de espera** en la cola procesos preparados.

- **Algoritmo de colas múltiples con retroalimentación:**
  - **crea varias colas de procesos preparados** en donde **cada cola tiene un algoritmo de planificación** y los **procesos se van moviendo entre las diferentes colas hasta que se ejecutan**.
  - es una **<u>mezcla</u>** de todos intentando obtener el <u>mejor rendimiento de cada algoritmo</u>.



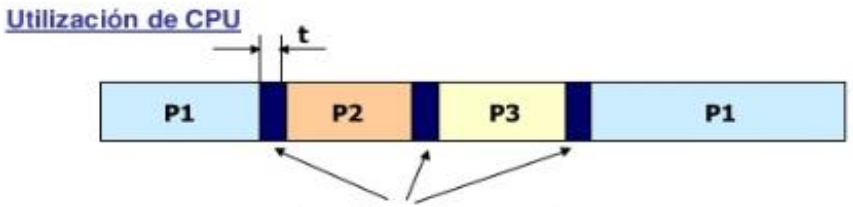
## 3.3.- Cambio de Contexto

Una cosa a tener en cuenta en los sistemas multitarea es el **tiempo que se tarda en realizar los cambios de contexto de los procesos**.

Un cambio de contexto es un **proceso del SO** que tiene que **guardar el estado del proceso que se está ejecutando y cargar el estado del proceso que se va a ejecutar.**

- el **tiempo** que tarda en realizar este proceso **afecta al rendimiento general del equipo**, si hay muchos cambios de contextos habrá **mucho tiempo de uso de la CPU** por parte del **SO y no** por parte de las **aplicaciones** de los usuarios.

Recordamos que el **estado** de un proceso se encuentra en su **BCP**. Cuando un proceso es interrumpido deberá **reanudar su ejecución exactamente por el mismo punto por donde lo dejó**, así que el SO toma su estado y lo guarda. Luego carga el BCP del proceso elegido por el planificador para que se ejecute.




$$\text{Utilización de CPU} = \frac{T(P1) + T(P2) + T(P3)}{T(P1) + T(P2) + T(P3) + 3t}$$

# 4.- Gestión de memoria.

La memoria es otro recurso clave que tendrá que gestionar el SO.

Para que un **proceso** se pueda ejecutar no sólo requiere tiempo de procesamiento sino también **estar cargado en memoria principal (RAM)**.

El software permanece en **memoria secundaria**, donde **no puede ser ejecutado**.
Para **ejecutar** y crear un **proceso** el SO debe asignar a ese programa un **espacio de direcciones de memoria**.
El encargado es el **administrador de memoria de los SO (Memory Management Unit MMU).**

Los SO Windows/Linux son sistemas **multitarea**, son capaces de albergar en memoria varios procesos en estado **listo**, que se irán ejecutando según el **algoritmo** de **planificación** del SO.

La **función principal** del gestor de memoria es:

- **asignar memoria RAM a los procesos** que la soliciten.

**Otras funciones** serán:

- **Control de memoria**: controlar las zonas de memoria **libres** y las **asignadas**, además de saber las zonas de memoria que **corresponden a cada proceso**.

- **Retirar** la memoria a los procesos (liberar) cuando terminen.

- **Reubicación**: como no se puede saber a priori en que zona de memoria se carga el proceso se realiza **direccionamientos relativos** para permitir que un **programa pueda ser cargado y ejecutado en cualquier parte de la memoria**.

- **Protección/Seguridad**: evita que **procesos cargados** en memoria **interfieran** unos con otros, accediendo a zonas de memoria que no les corresponden.

**Comprueba** que las **referencias a la memoria** generadas por un proceso en ejecución sólo **hacen referencia a la zona de memoria asignada** a ese proceso y no de otros procesos o SO.

- **Compartición**: puede ser necesario que **varios procesos puedan compartir y actualizar estructuras de datos comunes.**

*Ejemplo: un sistema de BBDD.*

- Control de flujo de información de **Jerarquía de memoria**: si disponemos de memoria **caché** más rápida que la RAM, en la que se mantienen los datos de acceso más frecuente.

- Gestionar el **trasvase de información** entre **memoria principal** y **secundaria** cuando la memoria RAM no sea suficientemente grande para acoger a todos los procesos. Los procesos pueden ser **intercambiados** a disco y más tarde, si es necesario, vueltos a cargar en memoria.

- **Controlar y evitar en lo posible casos de fragmentación** de la memoria.

Existen dos **tipos** de fragmentación de la memoria principal:

- fragmentación **interna**: sucede al **malgastarse el espacio interno de una partición cuando el proceso o bloque de datos cargado es más pequeño que la partición**.
- fragmentación **externa**: sucede cuando la memoria externa a todas las particiones se divide cada vez más y van quedando huecos pequeños y dispersos en memoria difícilmente reutilizables.

## 4.1.- Gestión de memoria en SO monotarea.


En sus orígenes los SO **no incluían ningún gestor de memoria**.
El **programador** tiene **control completo** sobre el espacio total de memoria.
La memoria se utiliza para almacenar el **proceso** en ejecución y el **SO**.
Los procesos se ejecutan **secuencialmente** a medida que van terminando los anteriores.
Esquema más **sencillo**, en cada momento la **memoria alberga un solo proceso y reserva otra zona de la memoria para el SO**.
Mecanismo de **protección** para evitar accesos a la parte del SO de los procesos de usuario.

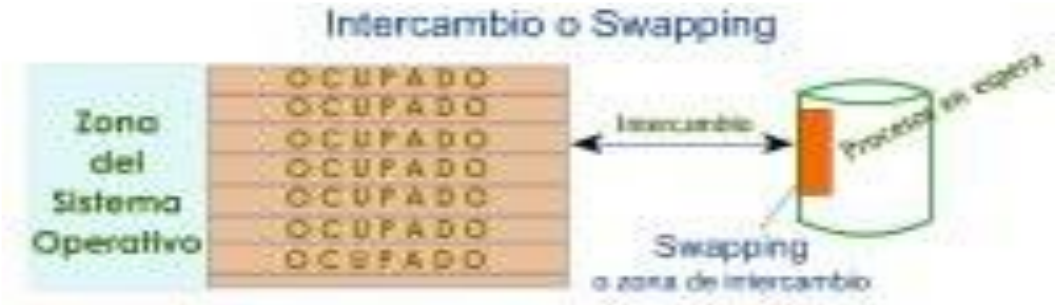
## 4.2.- Gestión de memoria en SO multitarea.

Los SO **actuales** son sistemas **multitarea** (varios procesos **simultáneos** en ejecución).
Estos **deberán estar también simultáneamente en memoria**.
Deberá existir mecanismos de gestión para **distribuir y proteger la memoria principal entre todos estos procesos** que quieren ejecutarse.

**Intercambio o swapping**

La RAM es un recurso **limitado**, puede ocurrir que haya **más procesos** esperando a ser cargados en memoria **que zonas libres en la misma**.

El SO **sacará de la RAM algunos procesos** y los llevará a un área de **disco** (memoria secundaria) conocida como área de intercambio (SWAP). Los procesos permanecerán allí hasta que **puedan ser recuperados de disco y reubicados en RAM**.

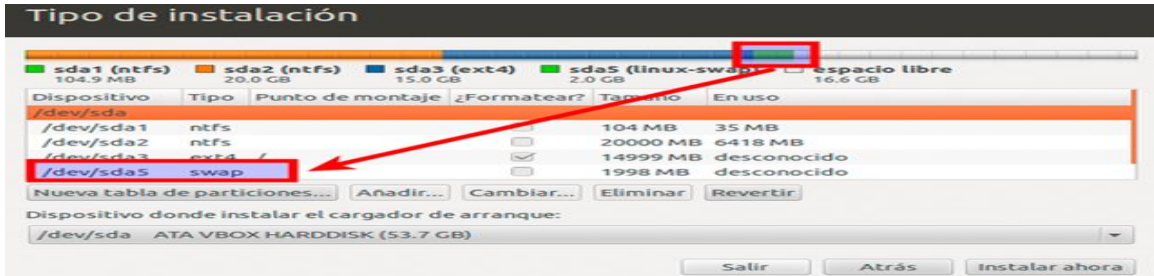



Los procesos **candidatos** a ser **expulsados** de RAM son los que están en estado **bloqueado** (al terminar su ráfaga de E/S pasará a **preparado**, no podrá pasar a **ejecución** de forma directa).

Cuando el proceso intercambiado termine su ráfaga de E/S y esté listo para ejecutarse pasa a un estado **listo en memoria secundaria** y cuando el gestor de memoria lo decida copiará toda la información del proceso en RAM y volverá al estado **listo para ser ejecutado**.

**Windows** la realiza con un **fichero** oculto **C:\pagefile.sys** cuyo tamaño lo controla el propio SO.

**Linux** con una **partición** que se crea en la instalación del sistema (tamaño 2 veces el de RAM).



# 4.2.1.- Asignación de particiones fijas.

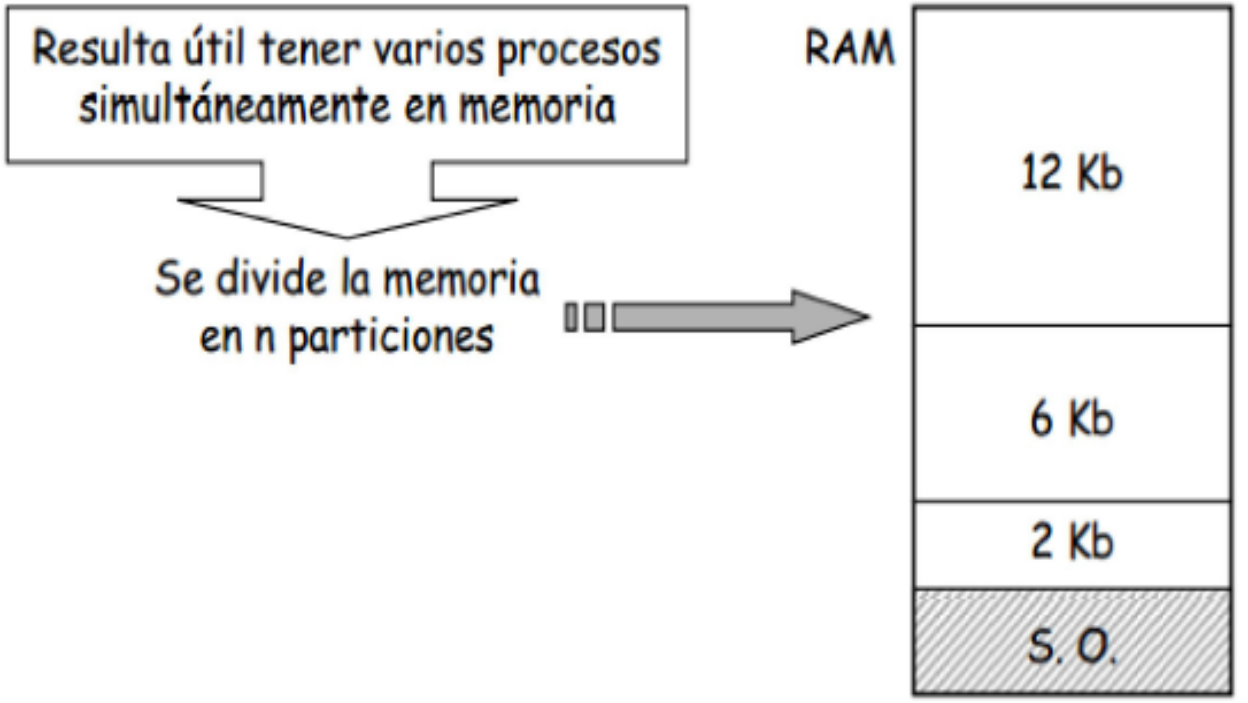
El gestor de memoria reserva espacio de memoria para el SO y el resto para los **procesos de usuarios**.

Cuando existen **varios procesos** que requieren ser cargados en memoria el gestor de memoria tiene que **organizar el espacio para ubicarlos**.

Hay varias **alternativas**, una de ellas es dividir el espacio de memoria en **particiones fijas**.
Estas particiones podrán ser:

- todas del **mismo tamaño.**
- tener **distintos tamaños**.

Se **establecen de forma lógica** por el SO y están **predefinidas** antes de que lleguen los procesos.
El **número** de particiones y el **tamaño** de cada una se mantiene **fijo en el tiempo**.



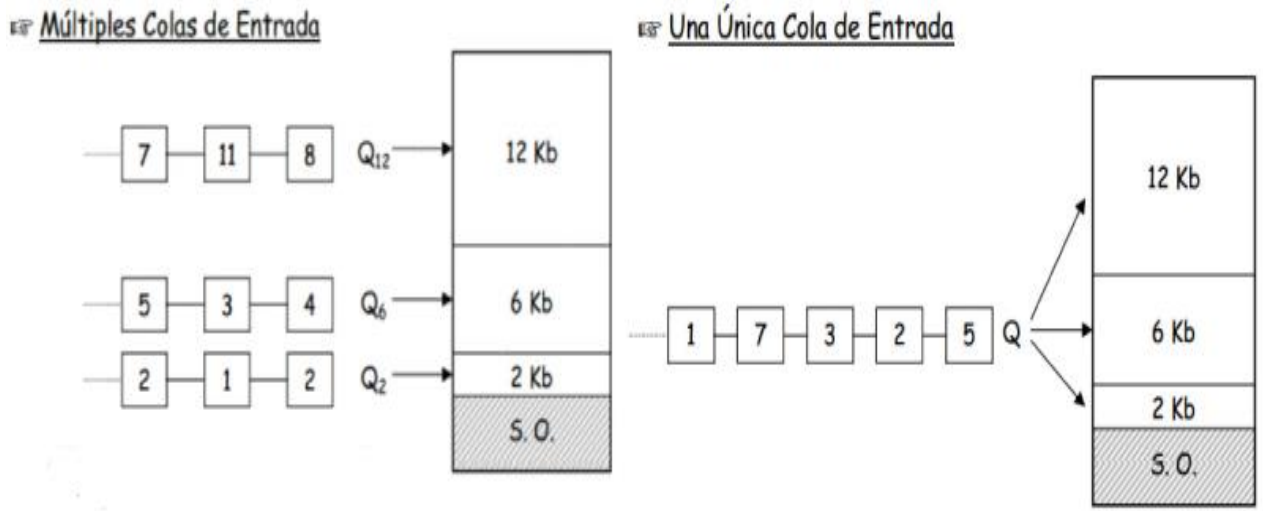
La **asignación** de particiones a procesos se hace de **dos formas**:

- **una cola por cada tamaño de partición**: se coloca cada trabajo en la **cola de la partición más pequeña** en que quepa dicho trabajo, a fin de **desperdiciar el menor espacio posible**.

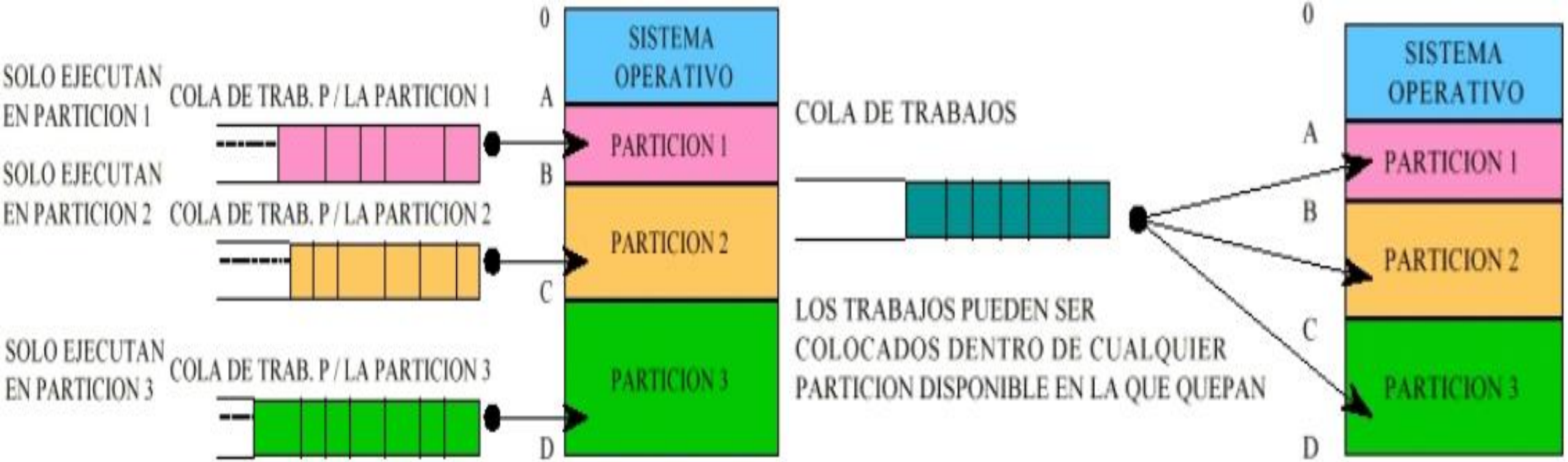
La **planificación** de cada cola se hace por separado y, como cada cola tiene su propia partición, **no hay competencia entre las colas por la memoria**.

**Desventaja:** cuando la cola de una partición grande está vacía y la cola de una pequeña está llena

- **una única cola común a todas las particiones**: el SO **decidirá** en que **partición se ubica cada proceso**, en función de disponibilidad de particiones y necesidades del proceso.




**NOTA**: El número de particiones distintas determina el **grado de Multiprogramación**




Cuando un **trabajo termina**, **libera** su región de memoria, que puede ser usada para llenarla con **otro trabajo de la cola**.

Una vez definidas las particiones, el SO necesita llevar la **cuenta de sus estados**: libre/en uso para propósitos de asignación.

El estado y los atributos de las particiones se recogen en una **estructura de datos llamada tabla de descripción de particiones (TDP).**

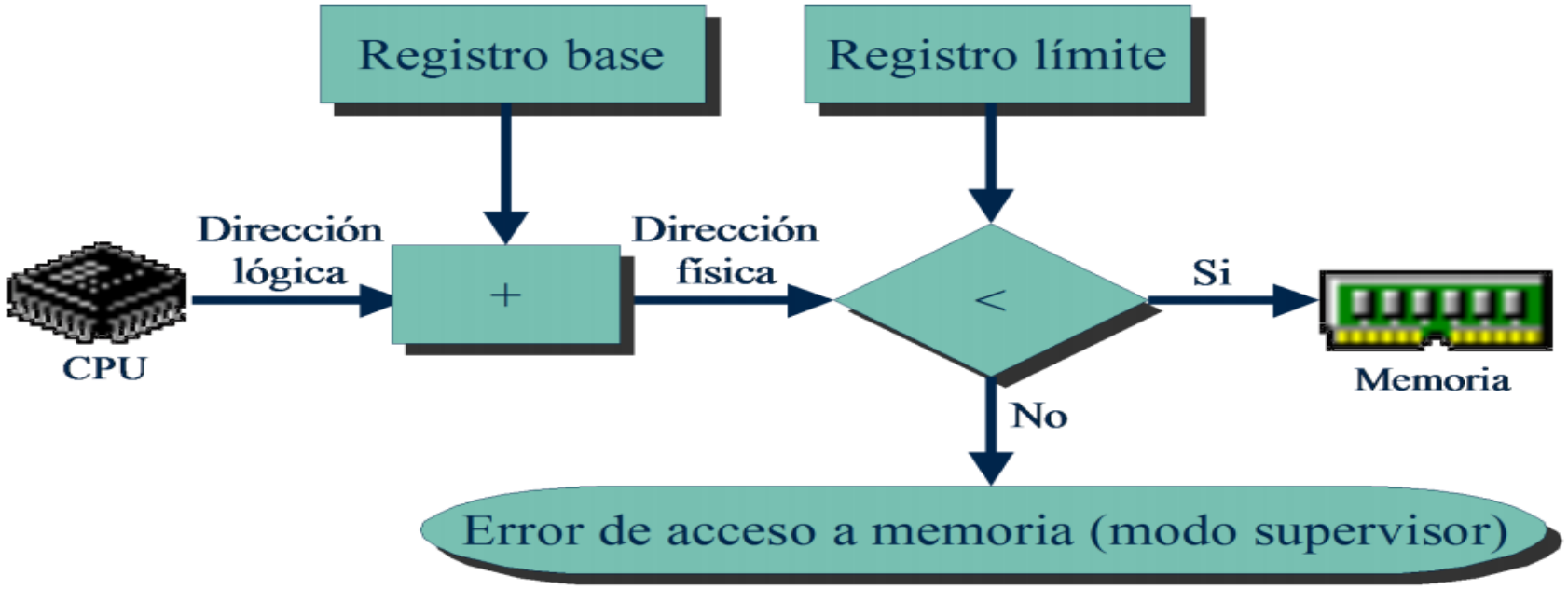
**Estrategias** de asignación de particiones:

- **Primer** ajuste
- **Mejor** ajuste
- **Peor** Ajuste

El **gestor de memoria establecerá mecanismos** para **impedir** que un proceso **pueda acceder a una zona de memoria que está fuera de la partición** que le corresponde.

Cada **partición** está descrita por:

- **dirección inicial (base)**
- **tamaño**
- **estado**



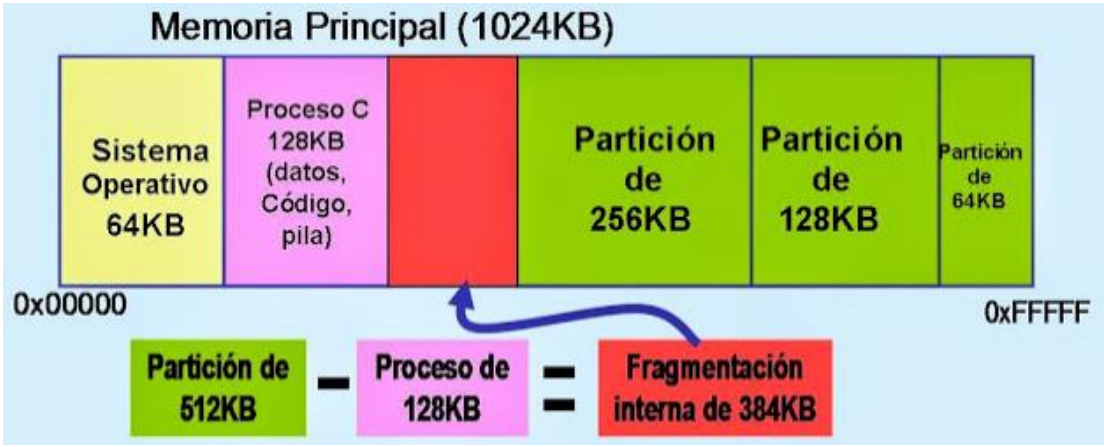

Los campos **base** y **tamaño** son **fijos**. Esto permite **proteger** el espacio de memoria que se encuentra antes y después de un programa en ejecución.

Existen **dos tipos de desaprovechamiento** de la memoria:

- Fragmentación **interna**: parte de memoria que no se está usando porque es interna a una partición asignada a una tarea.

*Ejemplo: Un trabajo que precise **m palabras** de memoria puede ejecutarse en una **región de n palabras**, donde n>=m. La diferencia entre estos dos números **(n - m) es la fragmentación interna**.*

- Fragmentación **externa**: ocurre cuando una partición disponible no se emplea porque es muy pequeña para cualquiera de las tareas que espera.




La selección de los tamaños de las particiones es un **compromiso** entre los dos casos de fragmentación.

Hay que encontrar una buena **relación** entre tamaños de partición y requerimiento real de memoria de los trabajos.

***La gestión de la memoria con asignación de particiones estáticas consiste en:***

- *La división de la memoria principal en partes fijas de igual tamaño.*
- *La división de la memoria principal en partes variables de igual tamaño.*
- *La división de la memoria principal en partes fijas de diferente tamaño.*
- *La primera y tercera respuestas son ciertas*

Ejercicios Memoria Particiones Fijas

# 4.2.2.- Asignación de particiones variables.

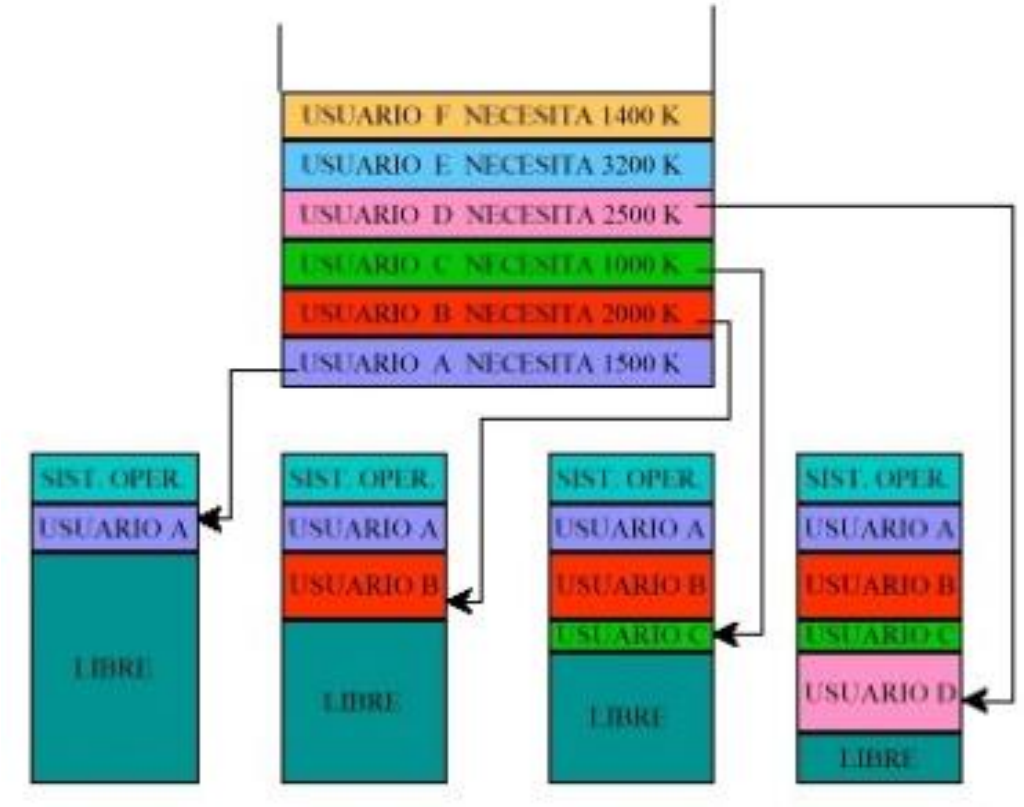
Las **particiones fijas** tiene la **desventaja** de no aprovechar todo el tamaño de cada partición, ya que el **proceso se adapta a los tamaños fijos ya preestablecidos en memoria**.

Alternativa: **asignación** de memoria a los procesos **mediante particiones variables**.
La idea es **crear las particiones dinámicamente**, conforme **llegan los procesos** y en **función de los tamaños de estos**.
Técnica **más realista** que **aprovecha mejor el espacio** de la memoria.

Este mecanismo se ajusta a la **realidad** de que el **número y tamaño de los procesos varía** dinámicamente.

No se está sujeto a un número fijo de particiones que Pudieran ser muy grandes o demasiado pequeñas.

**Ventaja**: consigue un **mejor uso de la memoria.**
**Inconveniente: mayor complejidad.**

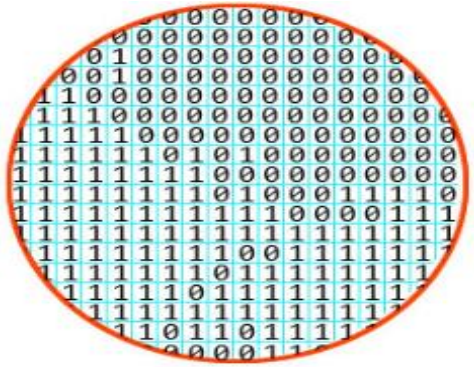


Cuando se **carga un proceso en memoria**, el gestor de memoria **crea** una partición adecuada que asignar al proceso en cuestión, es necesario **localizar un área libre** de memoria que sea igual o mayor que el proceso, si es así **se fabrica la partición.**

Inicialmente toda la memoria está a disposición de los programas de usuario, y puede considerarse como un gran bloque de memoria disponible, un **hueco**.

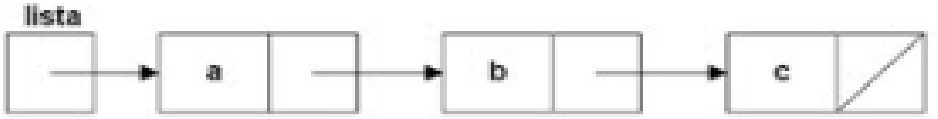
Problema de **mantenimiento de un registro de zonas libres y ocupadas** que sea **eficiente:**

- en **tiempo** para la asignación
- **aprovechamiento** de la memoria.



**Formas** de mantener este registro:

- **Mapa** de bits
- **Listas enlazadas** para las particiones libres y asignadas

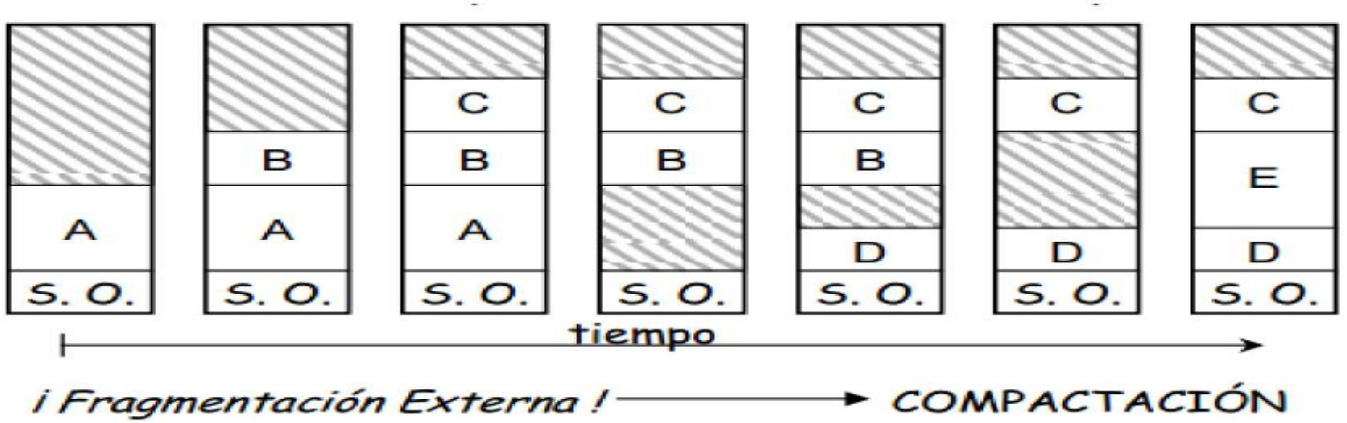



En la asignación de particiones variables, el **SO debe llevar el control** de qué partes de la memoria están **ocupadas** y **libres** (disponibles).

**Ventaja**: no existe o es muy pequeña la fragmentación interna.
**Inconveniente**: **fragmentación externa** o memoria desaprovechada entre particiones.
La memoria que debe ser asignada a un proceso nuevo debe ser **contigua**, la memoria libre disponible se halla dispersa debido a la fragmentación externa por lo que **no se puede usar**.

**Algoritmos** más habituales para la **selección de un área libre** de memoria a la hora de la creación de una partición son:

- **Primer** ajuste
- **Mejor** ajuste
- **Peor** ajuste




Si la **memoria resulta seriamente fragmentada**, la única salida posible es reubicar algunas o todas las particiones en un extremo de la memoria y así **combinar los huecos** para formar una **única área libre grande,** a este proceso se le llama **compactación.**
Como los procesos afectados deben de ser suspendidos y copiados realmente de un área de memoria a otra, es **importante decidir cuando debe realizarse** la compactación, ya que la solución de la compactación puede resultar un método de muy alto coste.

***Ejercicio de compactación***

*Calcular el **porcentaje de tiempo de UCP utilizado** para la compactación de la memoria en una máquina de 1Mb de memoria. La compactación se hace cada 0,5 seg. y se tarda 300 nseg. en copiar un byte. Los espacios desaprovechados en promedio son del 50% del tamaño de los bloques.*

*SOLUCION:*

*El espacio desaprovechado es del 50%, ello indica que se deben trasladar a un extremo de la memoria el 50% de la misma, es decir 0,5Mb.*

*Por lo tanto, el número de **bytes** que se debe trasladar es:*

$M = (1024 \times 1024) \times 0,5 = 524288$ *bytes*

*El **tiempo** total para el traslado es:*

$TCPU = (524288 \times 300) * 10^{-9} = 0,1573$ *seg*

| Submúltiplos | | |
|---|---|---|
| Valor | Símbolo | Nombre |
| $10^{-1}$ s | ds | decisegundo |
| $10^{-2}$ s | cs | centisegundo |
| $10^{-3}$ s | ms | millisegundo |
| $10^{-6}$ s | µs | microsegundo |
| $10^{-9}$ s | ns | nanosegundo |

*Si cada 0,5 seg se pasa 0,1573 seg compactando, entonces cada 1seg estará 0,3146seg, por lo que el 31,46% del tiempo de CPU se está desaprovechado en tiempos de compactación.*

*Obsérvese el elevado tiempo que se utiliza para las tareas de compactación.*

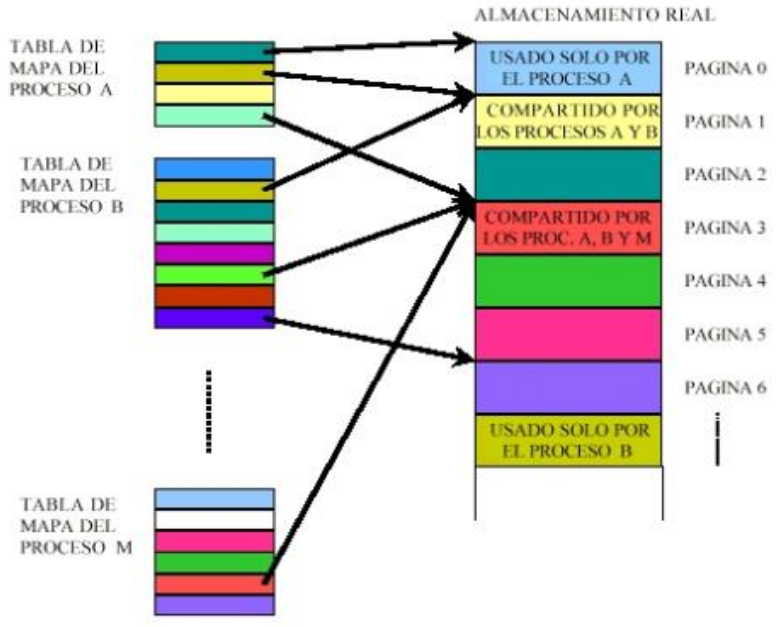
**Paginación.**

**OBJETIVO**: **suprimir** el requisito de la **asignación contigua de memoria física** a un programa.

Paginación **Pura**:

- Dividimos conceptualmente la **memoria física** en una serie de **porciones de tamaño fijo**, llamadas **marcos de página**.
- Dividimos el espacio de **direcciones virtuales de un proceso** en bloques de tamaño fijo del mismo tamaño llamados **páginas**.
- La **asignación** de memoria consiste **en hallar un número suficiente de marcos de página sin utilizar para cargar en ellos las páginas del proceso** solicitante.
- Las páginas **no tienen porque estar contiguas** en memoria.

**Ventaja:** reduce la **fragmentación externa** de la memoria principal.
**Inconveniente**: puede aparecer cierta **fragmentación interna**.



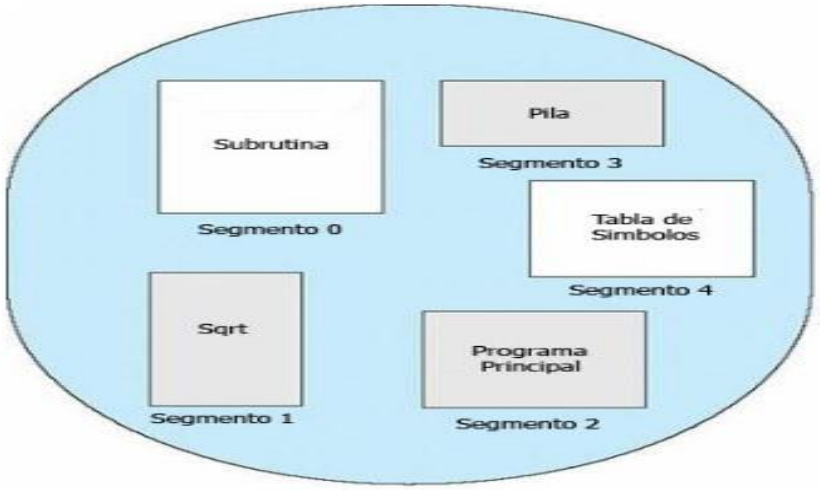
## Segmentación.

Cada proceso se **divide en una serie de segmentos**. La peculiaridad de estos segmentos es que su **tamaño no tiene que ser el mismo** y puede variar hasta un límite máximo.

- **Segmentación Pura:**

La idea es **dividir el espacio de direcciones** de un **proceso** en bloques que puedan ser colocados en áreas no contiguas de memoria. Un proceso se carga **situando todos sus segmentos en particiones dinámicas** que **no tienen que estar contiguas** en memoria.

El programa de usuario se **compila** y el compilador construye automáticamente los segmentos de acuerdo a la estructura del programa
Mediante la **agrupación de elementos** relacionados **lógicamente**.
*Ejemplo: pila, datos, código.*



Espacio de direcciones Lógicas

| | Limite | Base |
|---|---|---|
| 0 | 1000 | 1400 |
| 1 | 400 | 6300 |
| 2 | 400 | 4300 |
| 3 | 1100 | 3200 |
| 4 | 1000 | 4700 |

Tabla de segmentos

Memoria Física

Ejercicios Memoria Particiones Fijas y Variables

## 4.2.3.- Memoria virtual.

Hasta este momento los procesos se cargaban enteros en la RAM, pero puede suceder que:

- existan **procesos grandes que no quepan en memoria** y no puedan ser cargados
- queramos **tener más procesos** de los que físicamente caben.

La **memoria virtual** permite **dividir** los procesos en varias **partes** y **cargar sólo algunas** de ellas en memoria. Se basa en el uso de las **técnicas de paginación o segmentación**.

Es pues, un **esquema** de gestión de memoria donde puede que **sólo una parte del espacio de direcciones virtuales** de un proceso esté **cargada realmente en memoria física.**
En decir, permite la ejecución de procesos **parcialmente cargados.**

La técnica de paginación se basa en la **forma que tienen los procesos de ejecutarse**.
Los **programas** son muy grandes y realmente no pueden estar completos en memoria, pero es que **no hace falta**. Los programas cuando se **ejecutan** van **usando solo pequeñas partes de código** en un momento concreto.

***Ejemplo**: la **construcción de una casa**.*
*Supongamos que somos una **empresa de construcción** y vamos a construir un **edificio**.*
*Vamos a **necesitar** encofradores, pintores, albañiles, fontaneros, electricistas, etc…*
*El primer día de construcción ¿voy a **contratar a todos** al mismo tiempo? NO*
*No necesito un pintor hasta que las paredes nos estén levantadas, etc.*
*La construcción del edificio pasa por diferentes **fases** y en cada fase necesito unos especialistas y los voy contratando **conforme los vaya necesitando**.*

Los programas son divididos en partes por el SO y solo se van a cargar en RAM las partes necesarias, no tienen porque estar en memoria al 100%, el proceso solo usa las instrucciones que necesita.

Para saber qué parte está usando un proceso y saber que partes de un proceso se ha cargado en memoria existe una estructura en el **BCP** que es la **Tabla de Páginas**.
Cada proceso tiene su tabla de página en su BCP con la que el **SO asocia cada página del programa en qué zona de memoria principal se encuentra (marco de página).**

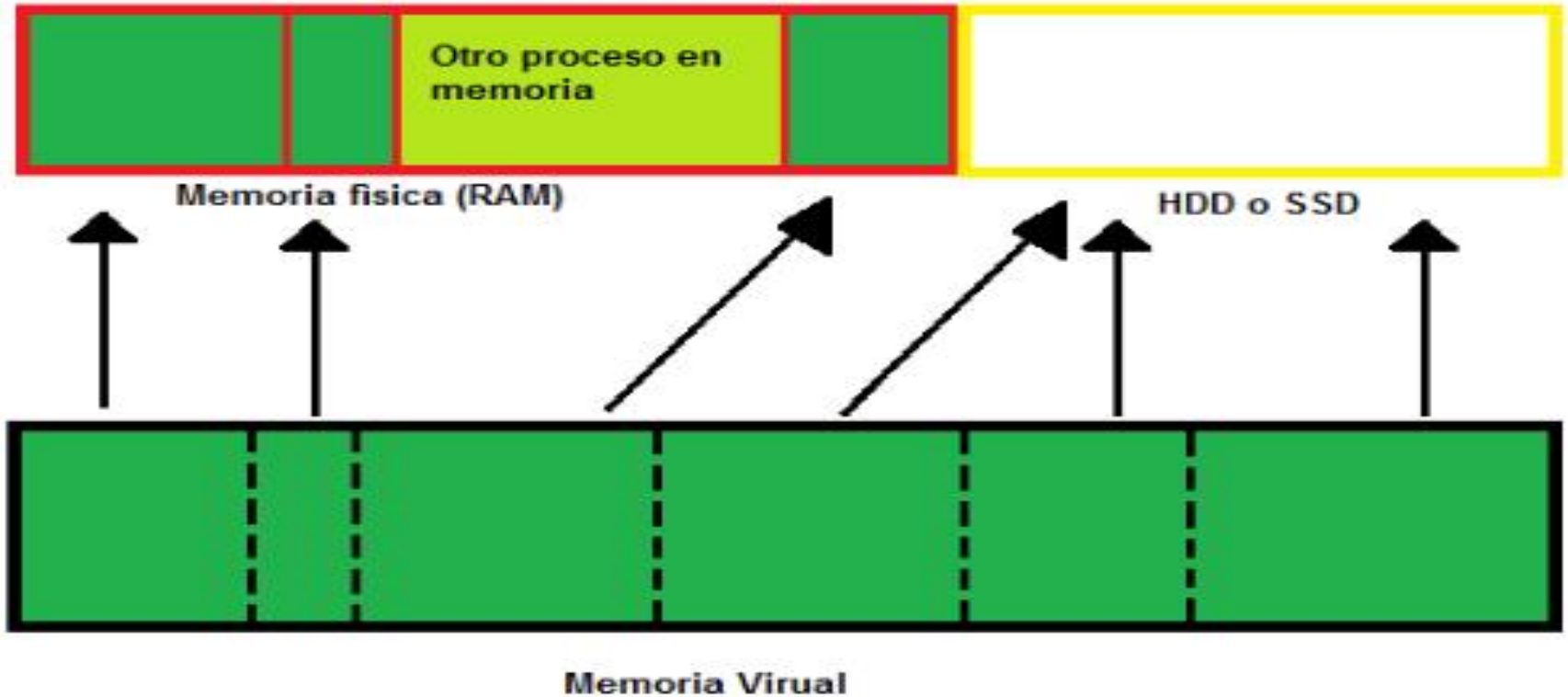
Así pues los programas se van cargando en bloques de tamaño fijo en memoria principal.

**El espacio de direcciones virtuales completo de un proceso reside en memoria secundaria**, y **trae a memoria principal** parte de ese espacio **cuando sea necesario**.

*Para el usuario, el programa estará realmente cargado en RAM,* ***aunque sólo se carga la parte del programa que en realidad se está ejecutando en ese instante****, el resto del programa en ejecución permanece temporalmente almacenado en disco para su posterior utilización si fuera necesario.*

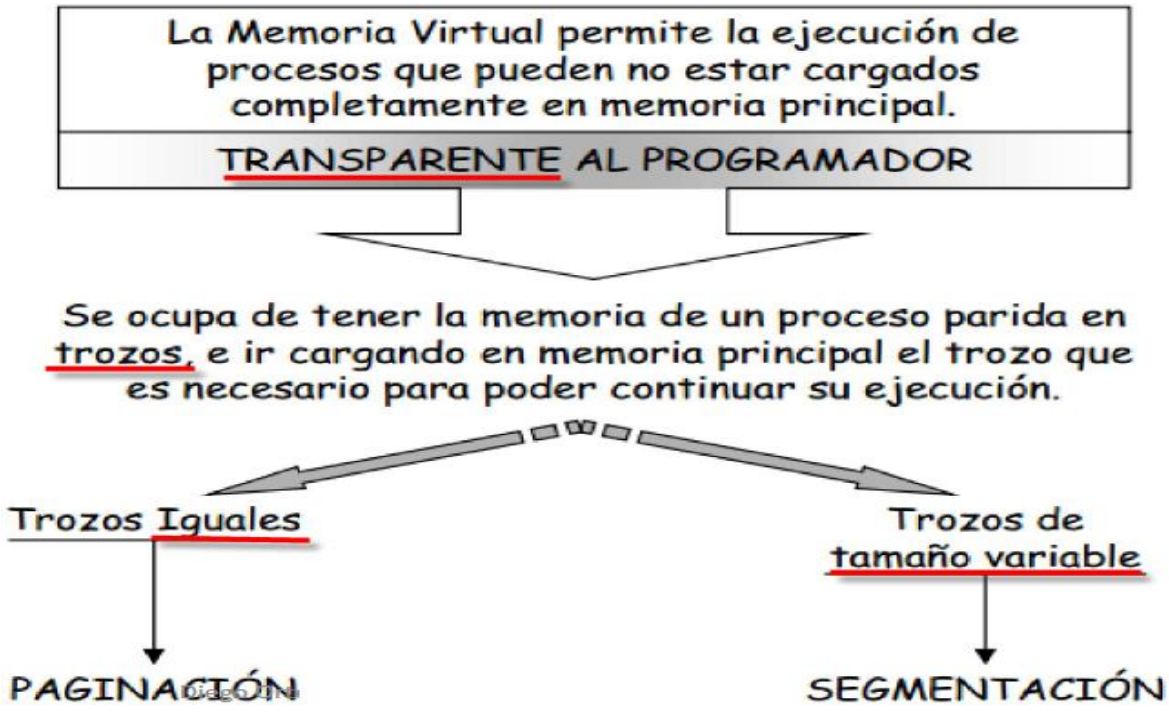
Es **tarea** del SO la **elección de:**

- **que sección traer**
- **cuando**
- **donde**



**Ventajas:**

- los **programas de usuario pueden ser mayores** que la memoria física. Un programa no quedará limitado por la cantidad de memoria física real. Los programadores podrán **escribir programas** para espacios de direccionamiento virtual **grande**.
- si cada usuario ocupa menos memoria física, más usuarios podrán **ejecutar al mismo tiempo más procesos** > mayor incremento de **productividad** y utilización de CPU.




**Inconvenientes:**

- la memoria virtual **no es fácil** de implementar y puede **reducir** sustancialmente las **prestaciones** si se usa inadecuadamente.

**Implementaciones**:

- **Memoria Virtual Paginada (se basa en paginación).**

Funcionamiento de la paginación simple, pero **no es necesario cargar todas las páginas** de un proceso para que éste pueda ejecutarse.

- La **memoria física se fragmenta** en bloques/particiones de tamaño fijo llamados **frames** o **marcos de página**.
- La **memoria lógica**(programas) también se **fragmenta en bloques del mismo tamaño llamados páginas**.
- Cuando hay que ejecutar un programa, se cargan sus páginas en cualquier frames que haya disponibles y se define la **tabla de páginas**.
- Las páginas **no** tienen porque estar **contiguas** en memoria.

Las páginas que **no se encuentren en memoria y se necesiten, las traerá posteriormente a memoria** el SO de manera automática.
El **tamaño de** página viene definido por el hardware (tamaño de la memoria, anchura del bus de direcciones, etc).

- **Memoria Virtual Segmentada (se basa en segmentación).**

Funcionamiento de la segmentación simple, pero **no será necesario cargar todos los segmentos de un proceso** para que este pueda ejecutarse.

  - Cada **proceso se divide en una serie de segmentos**, cuyo **tamaño no tiene que ser el mismo y puede variar hasta un límite máximo**.
  - Cada **segmento** tiene un **número** y una **longitud**, soportando la separación de la **visión** que el programador tiene de la memoria física al dividir los segmentos en **grupos lógicos**.
  - Un **proceso se carga situando todos sus segmentos en particiones dinámicas** que **no** tienen que estar **contiguas** en memoria.
  - Las **direcciones** especifican tanto el **número del segmento** como el **desplazamiento** dentro del segmento.

Si se necesitan segmentos no asignados en memoria se **traerán en el momento en que sean referenciados**.

Los **algoritmos** son bastante más **complejos** que los de la paginación.

- **Sistemas Combinados: Paginación segmentada y Segmentación paginada.**

**Funcionamiento de la Memoria Virtual**:

En Memoria Virtual hay que **separar**:

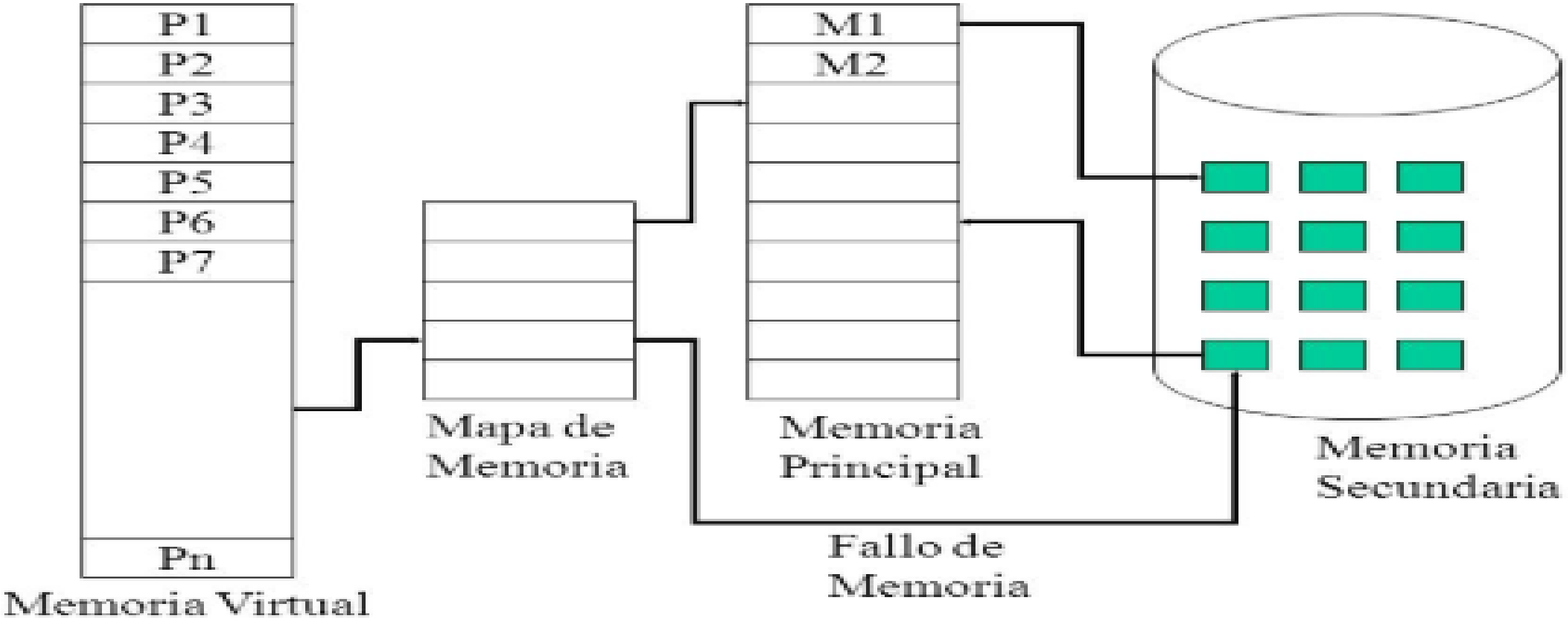
- conjunto de las **direcciones virtuales** (salen de CPU).
- **direcciones físicas o reales** (las que corresponden a memoria RAM).
- **mecanismos software y hardware** para **traducir cada dirección virtual en su correspondiente dirección real**.

Si la **dirección real** de un dato **no corresponda a memoria principal, sino a memoria secundaria**, habrá que **traer** dicho dato a RAM para que la CPU pueda continuar su ejecución.

La tarea del **hardware de traducción de direcciones en sistemas virtuales** es detectar si el **elemento referenciado esta en memoria real o no**.

Implementación: se **añade un bit de presencia**, a cada entrada de la Tabla de Proceso.

En el caso de que en el momento de la **traducción** el bit esté a **cero**, el hardware genera una **excepción** de **fallo de página** para anunciar el hecho al SO.



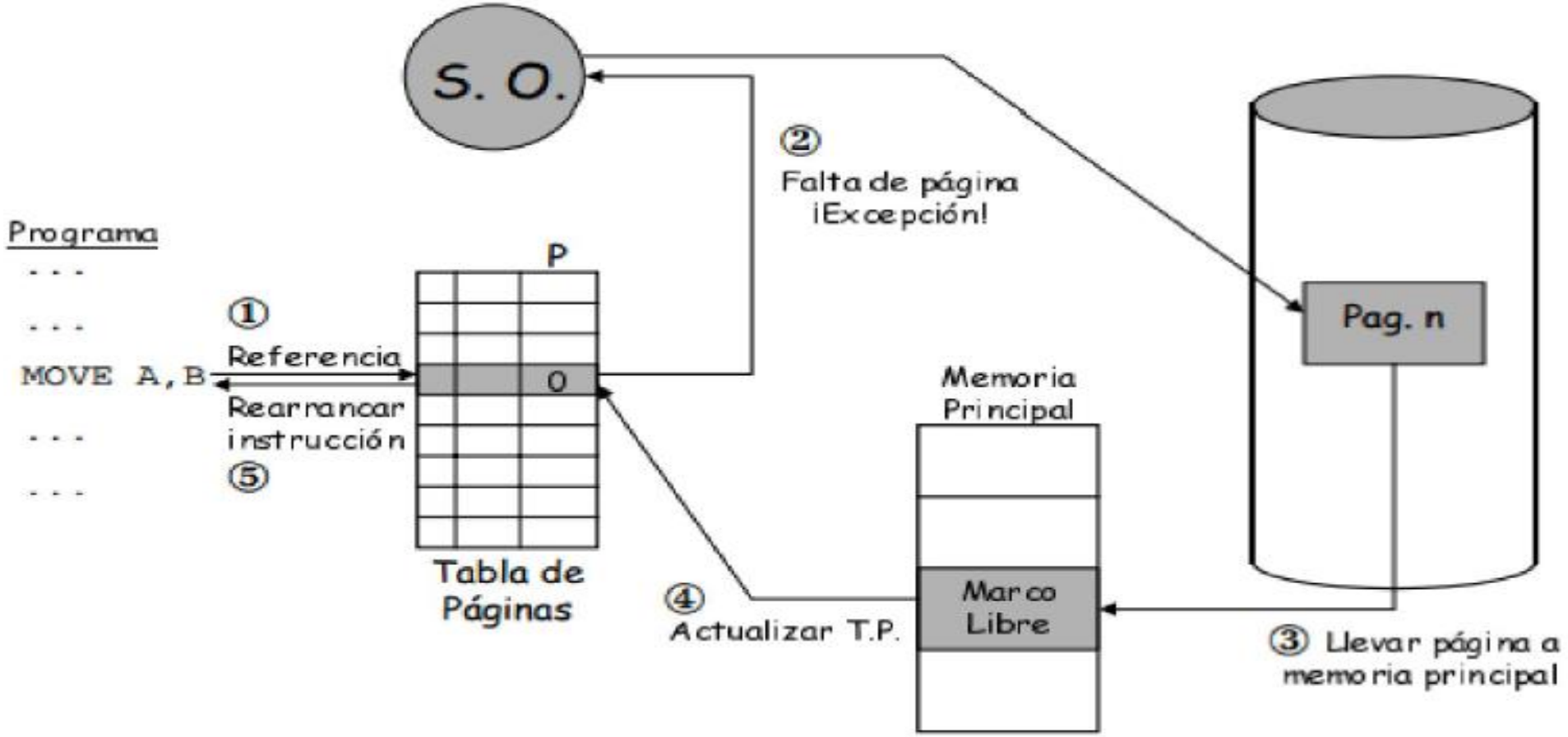

Un fallo de página se produce cuando: “en la **tabla de páginas del proceso no hay asociado un marco de página** a la página que se **quiere acceder”**.

El proceso en **Ejecución** queda **Bloqueado** hasta que la página que falta sea incorporada en RAM
**Procedimiento** a seguir es el siguiente:

- Se **verifica si la dirección es válida**. Si la dirección fuera **inválida** se enviará una señal al proceso o se abortaría. Si es una referencia **válida** y está en memoria se ejecuta la instrucción.
- Si es una referencia válida, pero **aún no se ha traído esta página** (bit de presencia cero), el SO detecta un **fallo de página y determina la página virtual requerida para traerla**. La instrucción queda **interrumpida** y se guarda el estado del proceso, para poder continuarlo en el mismo lugar y estado > **BLOQUEADO**
- Se **selecciona un marco libre**. Si **no existe** un marco libre, se tendría que ejecutar un **procedimiento para enviar un marco a memoria secundaria.**
- Cuando el marco queda libre, el SO examina la dirección en el disco donde se encuentra la **página necesaria y planifica una operación de lectura de la misma**. Mientras se carga la página, el proceso sigue **suspendido** y se permite ejecutar otro proceso.
- Cuando se completa la lectura del disco, la **tabla de páginas se actualiza** para indicar que ya se dispone de la página en memoria.
- La instrucción que produjo el fallo regresa al estado de comienzo y **se planifica su ejecución**, pudiéndose acceder a la página como si siempre hubiese estado en memoria > **PREPARADO**

**Políticas** de la **gestión** de la Memoria Virtual:

- Política de **asignación**: que cantidad de **memoria se asigna** a un proceso > por **prioridades**
- Política de **ubicación**: en que parte de la memoria se **coloca** el proceso, explicados en pag.70
- Política de **<u>sustitución</u>**: si no existe memoria libre cuando se debe añadir un nuevo elemento, hay que **desalojar** un elemento > **gestión fallo de página.**




Objetivo: **tasa de fallos de página lo más <u>baja</u> posible**.

**Algoritmos** de Sustitución Fallo de Página:

- **FIFO**: se sustituye la página que lleva más tiempo en memoria. No es necesario guardar el tiempo de entrada. Se crea una **cola**, según el orden de entrada, con todas las páginas de la memoria, cuando hay que **sustituir** una página, se elige la **primera** de la cola y la que se **trae** se inserta al **final** de la cola.
- **OPTIMO**: se sustituye la página que tardará más en volverse a utilizar. Algoritmo **irrealizable**, ya que el SO no sabe a priori que páginas se referenciarán en el futuro.
- **LRU**: se sustituye la página que no ha sido utilizada durante un periodo mayor de tiempo. Equivalente a aplicar el algoritmo óptimo hacia atrás en el tiempo.

  Dos posibles **implementaciones**:

  - Una **pila** con los números de las páginas, cuando una página se referencia, su número se coloca en la cumbre de la pila. Así, en la parte superior de la pila se tiene siempre la última página usada y en el fondo el de la página que hace más tiempo que se usó.
  - Mediante **registros contadores.**
- **Segunda Oportunidad**: **cola** circular y un **bit de referencia**, un puntero indica la página que se sustituirá a continuación, si se necesita un marco de página el puntero avanza hasta encontrar una página con **bit de referencia cero**, según avanza el puntero se ponen a cero el bit referencia.

Dependiendo de si ha **sido modificado o no el elemento retirado de memoria** puede tener que ser **escrito en disco o simplemente descartado**. Para ello se utiliza el **bit de modificación** o de suciedad en la Tabla de Procesos.

Un problema a tener en cuenta si se asignan pocas páginas a los procesos es el de la **hiperpaginación**, catástrofe o (thrashing)

- La explicación de esto es que con el alto grado de multiprogramación es imposible que todos los procesos mantengan suficientes páginas en memoria para que no se generen un gran número de fallos de páginas. Esto implica que se satura el canal de transferencia con el disco, que se bloquean muchos procesos esperando un intercambio de páginas y que el **procesador queda infrautilizado**

***Ejercicio Política de sustitución de Páginas.***

*Dada la siguiente cadena de referencia:*

*0 1 2 3 0 1 4 0 1 2 3 4*

*¿Cuántos **fallos de página** tendrán lugar si se dispone de 4 marcos de página inicialmente vacíos para el algoritmo **FIFO**?*

*¿Y si el algoritmo de sustitución fuese el **LRU**?*

*SOLUCIÓN:*

- *Para el algoritmo de sustitución **FIFO**: 0 1 2 3 0 1 4 0 1 2 3 4*

| 0 | 1 | 2 | 3 | 3 | 3 | 4 | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| x | 0 | 1 | 2 | 2 | 2 | 3 | 4 | 0 | 1 | 2 | 3 |
| x | x | 0 | 1 | 1 | 1 | 2 | 3 | 4 | 0 | 1 | 2 |
| x | x | x | 0 | 0 | 0 | 1 | 2 | 3 | 4 | 0 | 1 |

*Inicialmente la memoria está vacía luego las **cuatro primeras referencias causan fallos de página** que provocan que se traigan las cuatro primeras páginas*

*Las dos siguientes páginas que se solicitan, 0 y 1, si se hallan en memoria, por lo que no hay que hacer nada.*

*A continuación, se solicita la **página 4**, que no está en memoria, produciéndose, un **fallo de página**, para traer dicha página a memoria hay que buscar, inicialmente, la **pagina a sustituir**, dicha página es la de la **cabeza de la cola**, por ser la más antigua, la **página 0.***

*Las siguientes referencias a páginas también producen fallos de página, porque no se hallan en memoria, las páginas a sustituir serán las de la cabeza de la cola, por ser las más antiguas en memoria.*

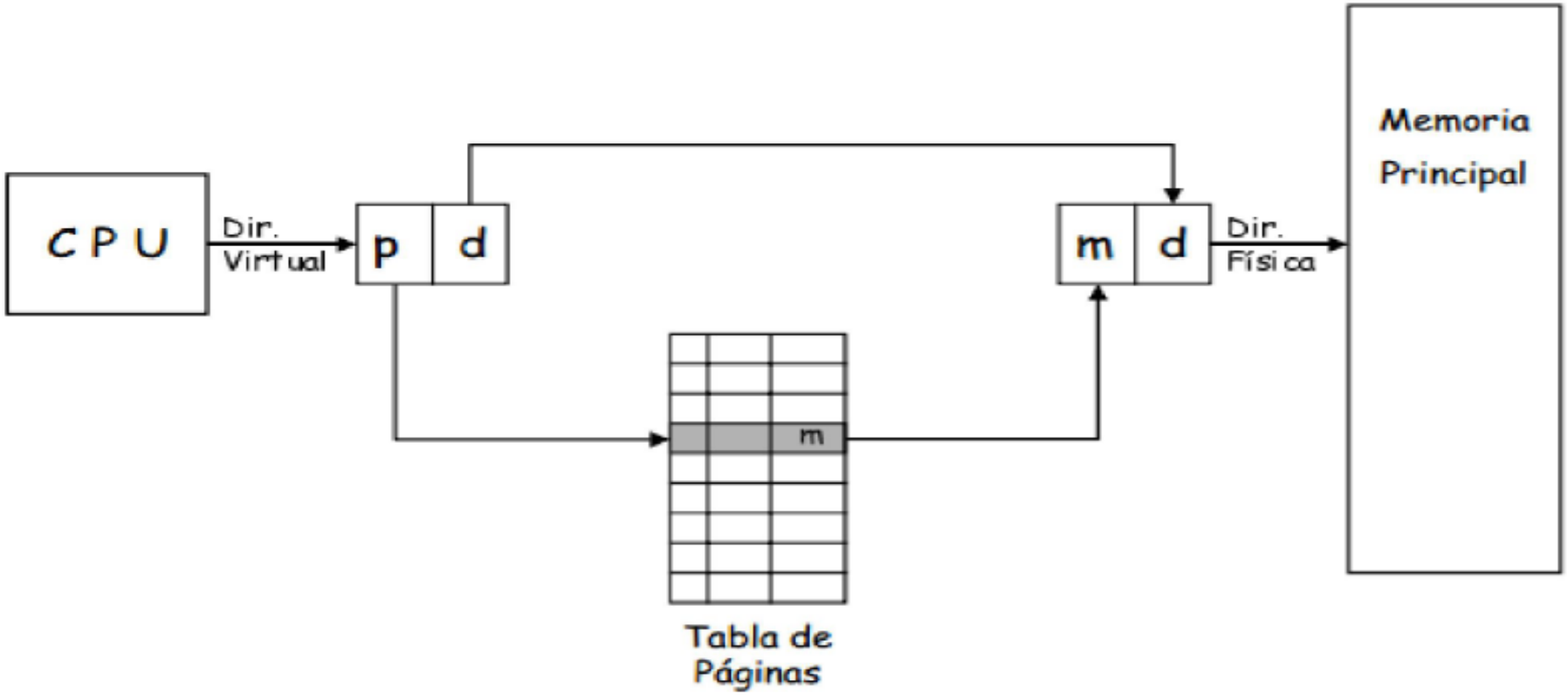
*Finalmente, tiene lugar **10 fallos de página.***

- *Para el algoritmo de sustitución **LRU**: 0 1 2 3 0 1 4 0 1 2 3 4*

| **0** | **1** | **2** | **3** | 0 | 1 | **4** | 0 | 1 | **2** | **3** | **4** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| x | 0 | 1 | 2 | 3 | 0 | 1 | 4 | 0 | 1 | 2 | 3 |
| x | x | 0 | 1 | 2 | 3 | 0 | 1 | 4 | 0 | 1 | 2 |
| x | x | x | 0 | 1 | 2 | 3 | 3 | 3 | 4 | 0 | 1 |

*Inicialmente la memoria está vacía, luego las **cuatro primeras referencias producen fallos de página**, por lo que hay que traer esas páginas de memoria.*
*Las referencias a páginas, 0 y 1, que se hallan en memoria, no producen fallos de páginas, pero se debe **mantener el orden en la pila**, por lo que cada vez que se referencia una página su número se elimina de la pila y se coloca en la cumbre. En la parte superior de la pila, se tiene siempre el número de la última página referenciada y en la parte inferior la página que hace más tiempo que se usó.*
*La referencia a la **página 4** que no se halla en memoria, genera un **fallo de página**, la página que se **sustituye** es la de la parte inferior de la pila, en este caso, la **página 2**.*
*Continuando con este procedimiento finalmente tiene lugar **8 fallos de página**.*

**Traducción** de Direcciones MV Paginada:




El componente de del SO que realiza la traducción de direcciones lógicas a físicas es la **MMU.**
La CPU emite **direcciones virtuales** (corresponden al **proceso en Ejecución**).
Una dirección virtual se **divide** en una parte de **entrada en la TP** y **desplazamiento** dentro de la dirección virtual.
La dirección física nos indica en la TP la **dirección en memoria principal**.
Añadimos a la dirección física el **desplazamiento** de la dirección virtual para obtener las **dirección física final**.

*Caso Práctico:*

*Tenemos un proceso cuyo espacio de direcciones virtuales se divide en cinco **páginas**:*

*000, 001, 002, 003 y 004*

*A cada página habrá que asignarle un **marco de página** de la memoria física.*

*Para la asignación se usa la **tabla de páginas**, que se **construye** cuando se carga el proceso y que contiene **tantas entradas como páginas tenga el proceso**, en este caso 5.*

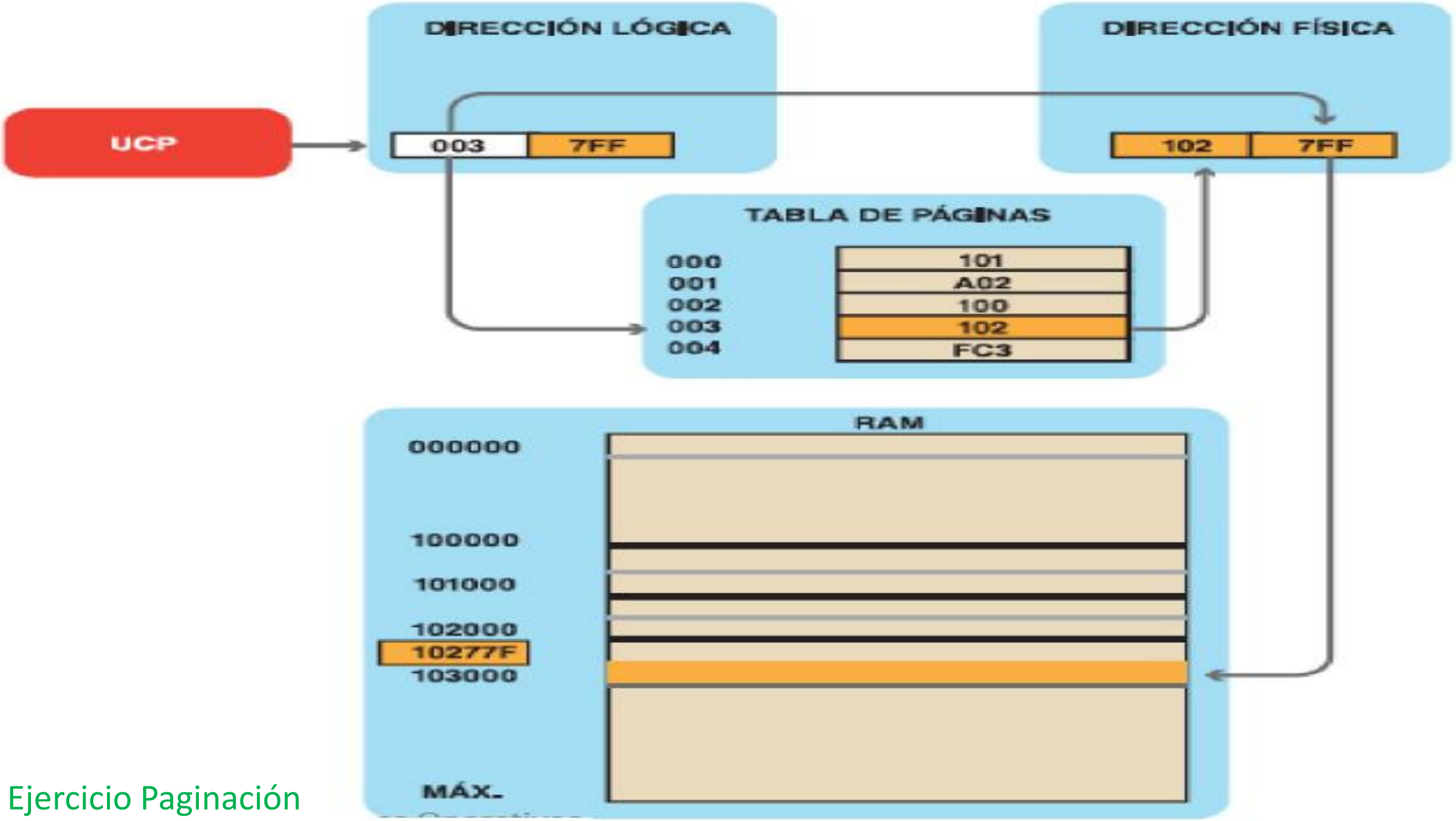
*Cada **dirección lógica** tendrá **dos** campos:*

- *El **primero**: accede a la página dentro de la tabla de paginas.*
- *El **segundo**: indica el desplazamiento que hay que realizar dentro de la página física para acceder a la información deseada.*

*Ejemplo:*

*La CPU indica como **dirección lógica 0037FF**. Esto quiere decir que debemos acceder a la cuarta página (003) de su tabla de páginas, donde está la **dirección del marco de página** de la memoria física con la información a ejecutar, que en este caso es la 102.*

*Una vez ubicados en dicho **frame**, debemos **desplazarnos** 7FF posiciones, con lo que se obtiene a partir de los dos campos la **dirección física a la que se quiere acceder**, en este caso es 1027FF.*




Ejercicio Paginación

# 5.- Gestión de la entrada/salida.

La gestión de los dispositivos de E/S o **periféricos**, corresponde a una parte del SO.

Existe una amplia variedad de dispositivos asociados a los sistemas informáticos cada uno de ellos con sus propios **requisitos electrónicos**. Son elementos **complejos** difíciles de usar directamente por los procesos.

La **memoria principal** está siempre preparada para entregar un dato cuya dirección le sea pedida por el procesador dentro de un tiempo casi **constante**, que dura unos pocos **ciclos de reloj** del procesador. Se puede decir que la memoria principal funciona **sincrónicamente** con procesador

Los **periféricos** operan de forma **asíncrona** con el ordenador, con **velocidades de transferencia** de datos muy variables, y no suelen presentar relación con el **reloj del procesador**.
La llegada de datos y los tiempos de transferencia de E/S son generalmente impredecibles, operan **asincrónicamente** con respecto al procesador. Todo esto hace que este control sea una **tarea difícil y compleja**.

Antes de obtener un dato de un dispositivo, el **procesador consulta** el estado del dispositivo utilizando **protocolos** de diálogo:

- si está **disponible**, el procesador adquiere el dato del controlador
- sino el procesador debe **esperar**, el SO debe elegir otro proceso a ejecución para **evitar que el procesador quede inactivo** mientras espera la terminación de las transacciones de E/S

Cuando la operación de **E/S se completa**, las señales de diálogo **alertan** al procesador sobre el hecho y hacen que el SO **reanude** la actividad correspondiente.

El SO se encarga pues de:

- que los **dispositivos se conecten al sistema y realicen sus funciones de forma eficiente**.
- **abstraer** la **complejidad** y peculiaridad **hardware** de cada periférico para que las aplicaciones de usuario puedan hacer uso de los periféricos de una manera **estandarizada** y más **sencilla**.
- actúa como **intermediario** por medio de los **controladores (drivers) del dispositivo**.

Una de las funciones del SO es **procesar la información** que obtiene de los periféricos y **mostrar** la **información** a través de los periféricos. Los periféricos se **clasifican** en:

- **Entrada**: reciben información y la transmiten al ordenador para su procesamiento.

*Ejemplo: el ratón, el teclado, el escáner, etc.*

- **Salida**: presentan la información procesada por el ordenador.

*Ejemplo: la impresora, el plotter (para impresión de planos y cartografía), el monitor, etc.*

- **Entrada y Salida**: aúnan ambas funciones.

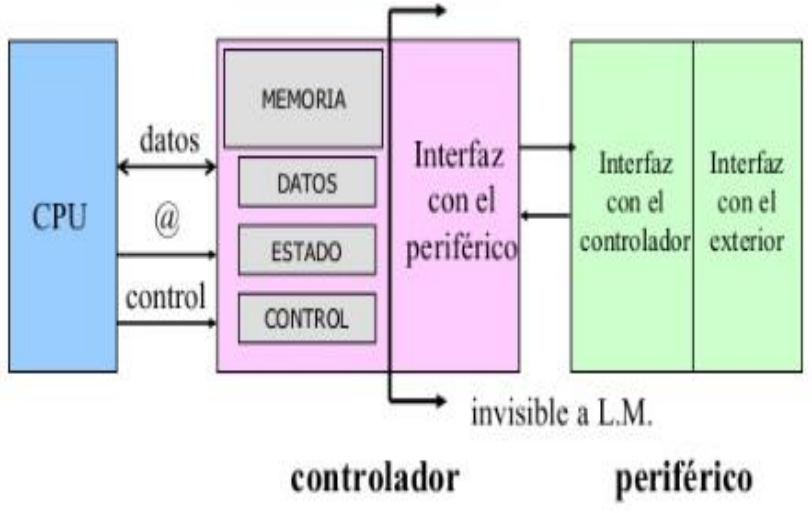
*Ejemplo: pantallas táctiles, **disco duro**, unidad de DVD, etc.*

## 5.1.- Controladores de dispositivo.

Para **entenderse** procesos con periféricos, los periféricos se dividen en **dos** partes:

- un **controlador (driver):** comunicación con la CPU.
- un **dispositivo mecánico**, **electromecánico** o **electromagnético**

Un **controlador** es un **software** suministrado por el fabricante del dispositivo o desarrollado por el SO, que actúa como **interfaz**
**Entre los programas de aplicación y el hardware.**
Permite **estandarizar** el acceso a los dispositivos.

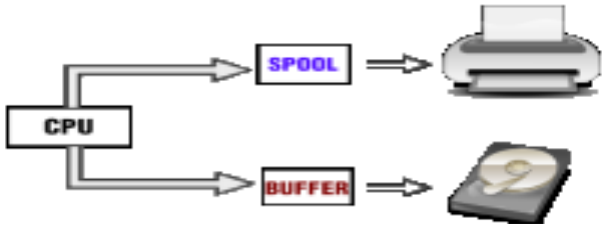


## 5.2.- Estructura de datos de la E/S.

La forma **que utilizan los periféricos** para manejar la información con la CPU son:

- **Spool**: los datos de salida se **almacenan de forma temporal** en una **cola** situada en un dispositivo de almacenamiento, **hasta que el dispositivo periférico requerido se encuentre libre**. De este modo se evita que un programa quede retenido porque el periférico no esté disponible. El SO dispone de **llamadas** para **añadir y eliminar archivos al spool**.

*Ejemplo: dispositivos que no admiten intercalación, como la impresora, ya que no puede empezar con otro hasta que no ha terminado.*




- **Buffer**: es para **dispositivos que pueden atender peticiones de distintos orígenes**. En este caso los datos no tienen que enviarse completos, pueden enviarse **porciones** que el buffer retiene de forma temporal. También se utilizan para **acoplar velocidades de distintos dispositivos**. Si un dispositivo lento va a recibir información más rápido de lo que puede atenderla se emplea un buffer para retener temporalmente la información hasta que el dispositivo pueda asimilarla.
  *Ejemplo: grabadora DVD y disco duro, la primera funciona a menor velocidad que el segundo.*

## 5.3.- Técnicas de la E/S.

Las distintas **formas de funcionamiento de la E/S** en los SO según la intervención de la CPU son:

- **E/S programada**: la CPU tiene todo el protagonismo, **inicia y lleva a cabo la transferencia.** Esta técnica **repercute en la velocidad de proceso del ordenador** porque la CPU debe dejar todo lo que está haciendo para ocuparse del proceso de entrada/salida.

- **E/S por interrupciones**: la CPU **ejecuta la transferencia pero el inicio es pedido por el periférico que indica así su disponibilidad**. Los dispositivos **avisan** a la CPU **cuando es necesario**. La idea es **reducir la inactividad** asignando algún otro trabajo al procesador mientras las operaciones de E/S activadas se encuentran en progreso. Requiere **asistencia hardware** en la sección de E/S, **señales** de diálogo y mecanismos de sincronización de sucesos tales como **interrupciones**.

- **Acceso directo a memoria (DMA):** la transferencia es realizada por un **controlador especializado**. Esta técnica **acelera** enormemente el proceso de la E/S y **libera** a la CPU de trabajo. Lo habitual es que los datos que se quieren escribir en el dispositivo o que son leídos del dispositivo provengan o vayan a la **memoria** del ordenador. La CPU **inicia** el proceso, pero luego este **continúa sin necesitar a la CPU**.

## 5.4.- Planificación de discos.

**Forma/orden en que el SO atiende las peticiones de lectura/escritura en disco**.
Utiliza **algoritmos de planificación del disco**.

El tiempo que tarda en **atenderse** una solicitud de L/E se desglosa en:

- Tiempo de **búsqueda**: para situar las cabezas en el cilindro al que se desea acceder.

Se compone de: Arranque, desplazamiento y detención.

- Tiempo de **latencia:** esperando a que el sector deseado pase por debajo de la cabeza.

Valor promedio: medio giro.

- Tiempo de **transferencia:** determinado por la tasa de datos del disco.

La **planificación** de disco busca reducir esos tiempos. Los algoritmos más habituales se centran en minimizar los **tiempos de búsqueda**.

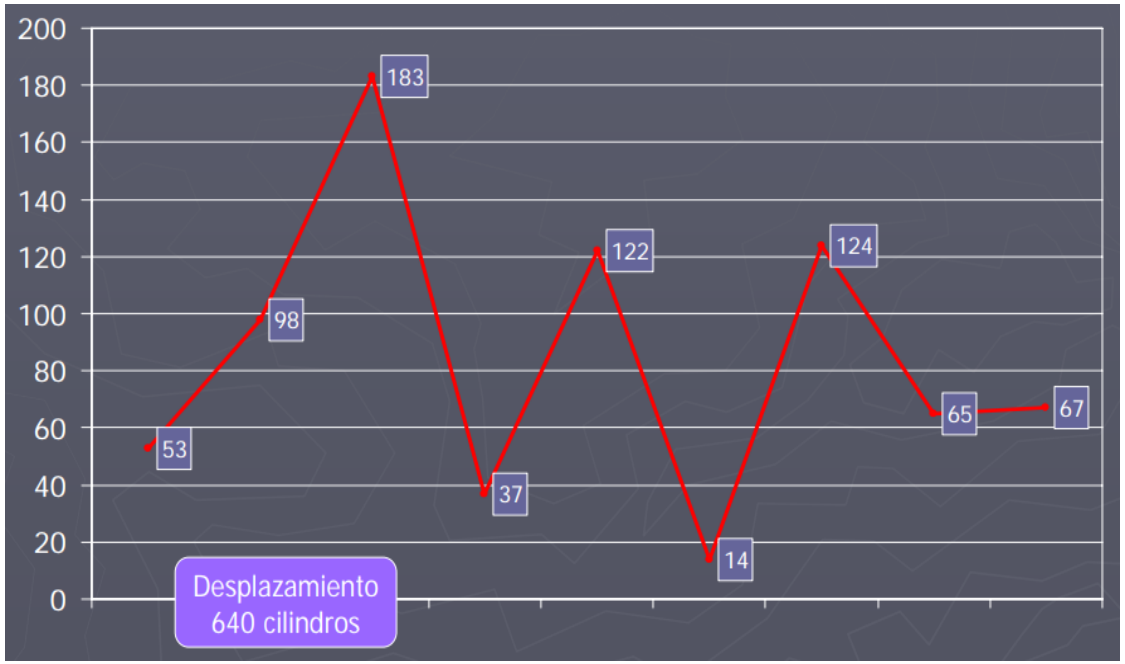
*Ejemplo:*
*Disco de 200 cilindros (200 pistas/superficie).*
*Cola de solicitudes a los cilindros 98, 183, 37, 122, 14, 124, 65 y 67.*
*Cabezas inicialmente posicionadas en el cilindro 53.*

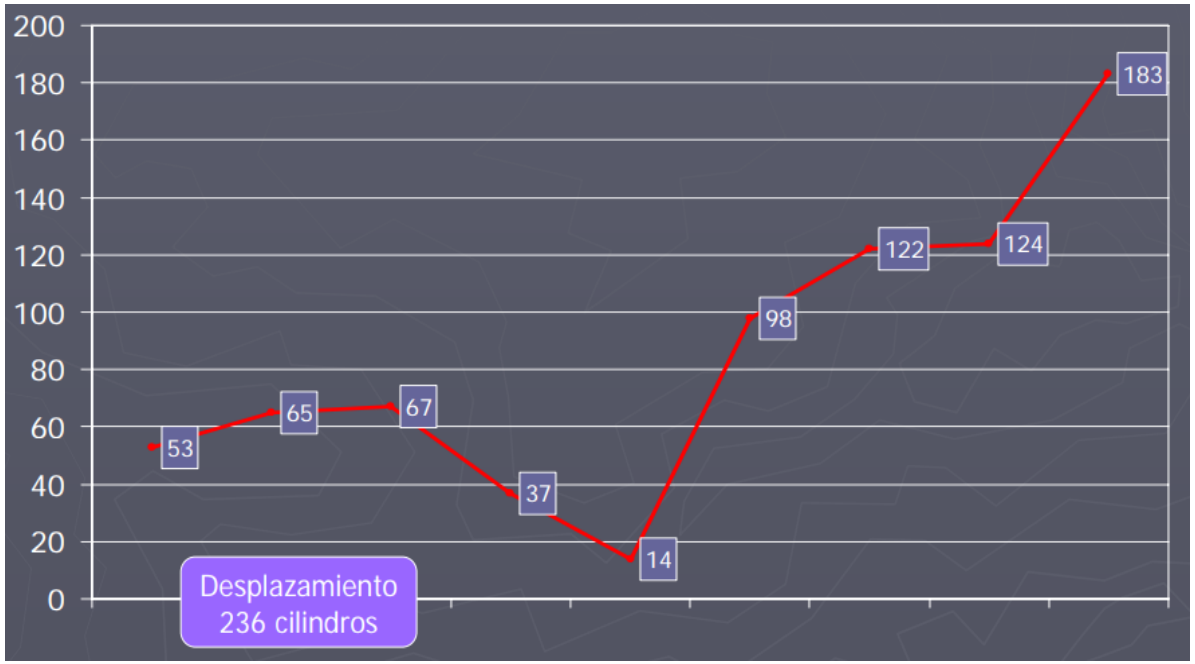
- **FIFO**:

Da servicio a las solicitudes de acceso a disco de la cola según el orden de llegada.

**Fácil** de programar, y **equitativo** en los tiempos de espera en cola.

Puede registrar **grandes desplazamientos** de las cabezas. Tiempos de espera **elevados**.




**FIFO** **SSTF**

- **SSTF** (Shortest Seek Time First, Primero la búsqueda más cercana) :

Atiende primero aquella petición que se encuentra mas cerca de la petición que se está procesando, aquella que **requiere el menor movimiento de la cabeza de L/E** desde su posición actual. Peticiones de L/E en zonas alejadas pueden sufrir inanición.

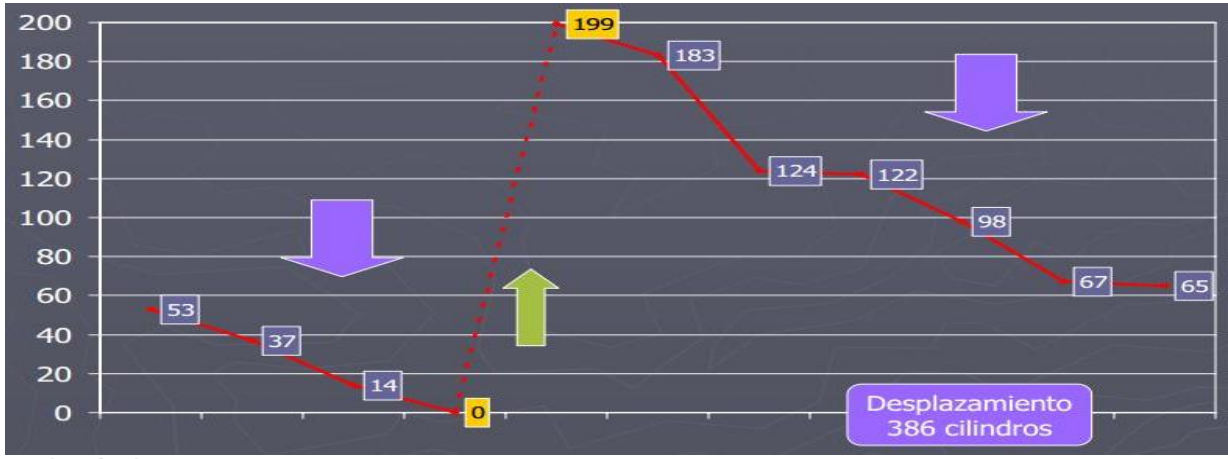
- **SCAN (ascensor)**:

Las cabezas se mueven de un extremo a otro del disco. Va atendiendo las solicitudes que va encontrando en el sentido en el que se van desplazando las cabezas de L/E por el disco. Cuando no hay más solicitudes en ese sentido, o se llega al extremo, se invierte el sentido para hacer lo mismo pero yendo hacia el otro lado.

Es necesario tener un **bit** que indique el **sentido** del movimiento.

Tiempos de servicio **acotados**, y más **variables** en los **extremos** que en el centro.


**SCAN**



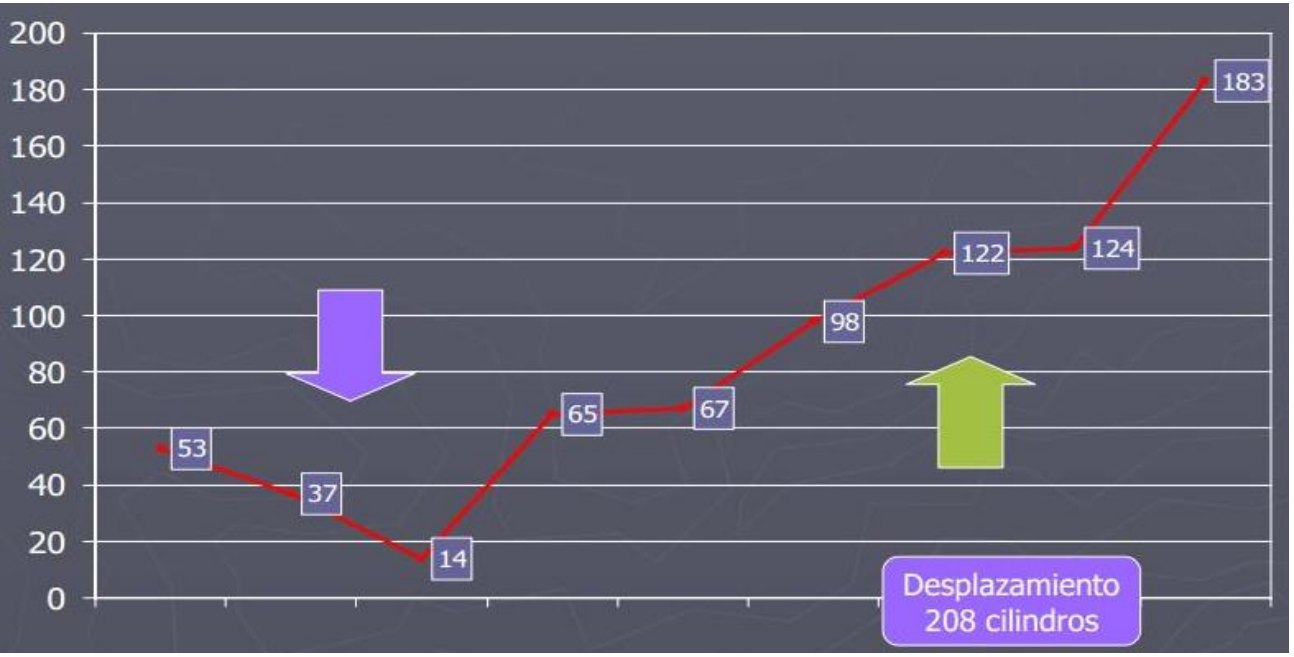

**C-SCAN**

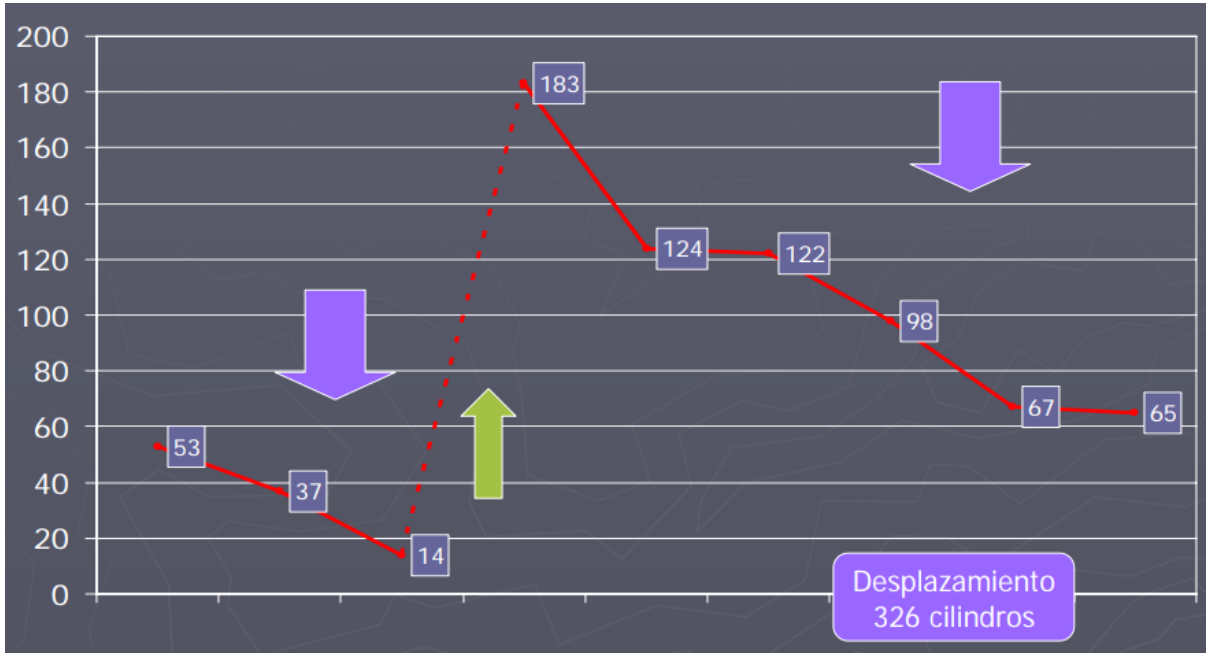
- **C-SCAN** (**Circular** SCAN):

Las cabezas se mueven del primer cilindro al último atendiendo solicitudes, y retornan al principio. Tiempos de **espera más uniformes**. El retorno consume relativamente poco tiempo, porque se hace sin paradas.

- **LOOK y C-LOOK:**

Las cabezas no se mueven hasta el extremo, sino **hasta la última solicitud pendiente** en el sentido del movimiento.




El algoritmo más empleado en sistemas de propósito general es C-SCAN.
Sistemas de tiempo real o multimedia requieren soluciones específicas.

Realizar Ejercicios Planificación de Disco

# 6.- Gestión del Sistema de Archivos.

Parte encargada de **gestionar los datos que residen en almacenamiento secundario.**

El sistema de archivos **provee** a los usuarios de las funciones para **operar con archivos y directorios** almacenados en disco proporcionando mecanismos de **protección y seguridad**.

**Fichero**: datos lógicamente relacionados ubicados en almacenamiento secundario que se organizan generalmente en colecciones caracterizadas por un nombre llamados **archivos**. Pueden contener información en formato texto, imágenes, bases de datos, etc.

El sistema de gestión de archivos **oculta** a los usuarios los aspectos relacionados con el dispositivo (hardware) dando una **abstracción** de un **espacio simple y uniforme de archivos con nombre**.

Los usuarios pueden apoyarse en un **conjunto único y uniforme de herramientas** para manipulación de archivos: crear, modificar, borrar, etc.

Cada SO utiliza **su propio sistema de archivos**, no obstante las **operaciones** que se pueden realizar sobre el sistema de archivos son bastante **similares**.

Los sistemas de archivos actuales utilizan **directorios** para organizar a los archivos mediante una **estructura jerárquica**.

**Objetivos** de un sistema de archivos son:

- **Optimizar** rendimiento mediante un **acceso rápido** a la información de los archivos.
- **Fácil actualización**: los cambios (añadir, borrar y modificar) no deben suponer una tarea complicada para el usuario y las aplicaciones.
- **Economía de almacenamiento**: intentar que los **archivos desperdicien la menor cantidad de espacio en disco posible**. Evitar **la fragmentación** de los discos.
- Tener **conocimiento** de todos los archivos del sistema.
- **Mantenimiento sencillo**: **ocultar detalles** y proporcionar **acceso estandarizado a los archivos**.
- **Fiabilidad**: asegurar la confianza en los datos, asegurar que los **datos escritos o leídos sean correctos y fiables**. **Minimizar** o eliminar la posibilidad de **pérdida o destrucción de datos**.
- Contabilizar el espacio de disco no utilizado por medio de un **depósito de bloques libres**.

- Control de **concurrencia**: controlar y asegurar el **acceso correcto a los archivos por parte de varios usuarios a la vez**, **bloqueando** el archivo en uso hasta que termine la operación de modificación en curso.
- Incorporar **mecanismos de seguridad y permisos**: en SO multiusuario proteger los archivos de un usuario del **acceso de los demás usuarios a través de los atributos** (permisos de escritura, lectura o ejecución).

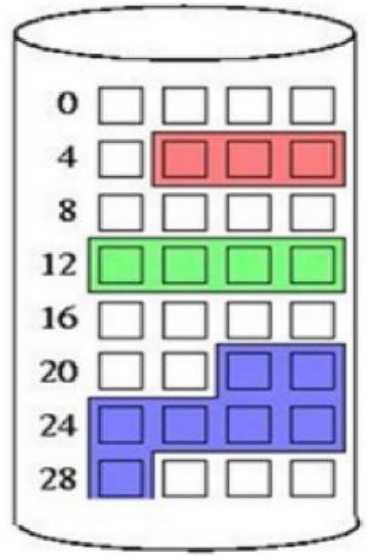
**Atributos**: constituyen **información adicional** con la que cada archivo queda caracterizado, indican el nombre, fecha de creación, tamaño, protección, contraseña de acceso, etc.

- **sólo lectura**: el archivo se puede leer pero no se puede modificar.
- **oculto**: el archivo existe pero no se ve.
- **modificable**: si es susceptible de modificarse o no.
- **sistema**: en el caso de pertenecer al propio SO

**Tipos de asignación de espacio en disco:**

- **Asignación <u>contigua</u>:** asignar **áreas contiguas** de disco en respuesta a peticiones en tiempo de ejecución, igual que se hace con la asignación de memoria principal.

**Ventaja**: acceso **rápido** a archivos sin necesidad de acceso a discos intermedios para localizar bloques deseados.
**Desventaja**: **fragmentación interna y externa** por lo que se **desaprovecha** disco.



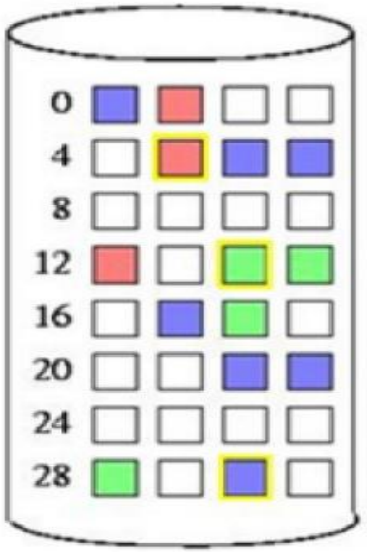

Directory:

| file | start | length |
|---|---|---|
| moo | 5 | 3 |
| fall | 12 | 4 |
| snow | 22 | 7 |

- **Asignación <u>no contigua</u>** de espacio de disco, dos **estrategias**:
  - **Encadenamiento:** lista enlazada.

  Sirve apara las listas de <u>archivos</u> y las de espacio <u>libre.</u>
  En los archivos, el directorio puede contener «el **puntero**», la dirección del **primer bloque de disco**.

  **Ventaja**: **simplicidad, no** produce **fragmentación externa**.
  **Desventaja**: acceso **lento** a archivos, **fragmentación interna**.

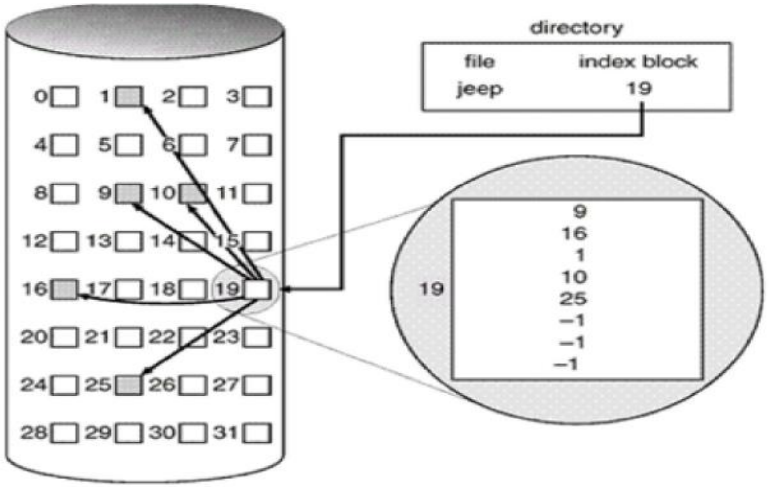



Directory

| File | Start |
|---|---|
| moo | 5 |
| snow | 30 |
| fall | 14 |

| | | | | |
|---|---|---|---|---|
| 0 | EOF | EOF | | |
| 4 | | 12 | 0 | 17 |
| 8 | | | | |
| 12 | 1 | | 18 | EOF |
| 16 | | 22 | 28 | |
| 20 | | | 6 | 7 |
| 24 | | | | |
| 28 | 15 | | 23 | |

- **Indexación**: se almacena la **dirección del bloque índice que contiene punteros a los bloques de datos del archivo en cuestión**.

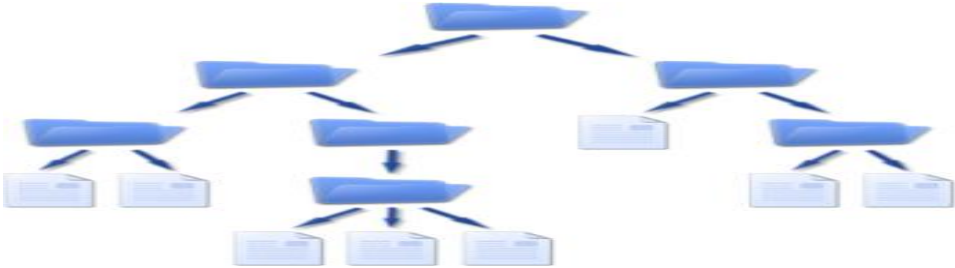



**Ventaja**: no **fragmentación externa** y eficacia del **acceso aleatorio**.

**Desventaja**: acceso a disco necesario para obtener la dirección del bloque deseado en disco.

**6.1.- Organización lógica** (software) **y física** (hardware)**.**

- El **nivel físico** de almacenamiento de datos en un disco duro consiste en el **formateo** en pistas, sectores, cilindros y platos. Es muy dependiente del hardware concreto que se esté usando y además **funciona a muy bajo nivel**.
- El **nivel lógico,** los sistemas de archivos proveen una **capa de abstracción que oculta los detalles** puramente **hardware** al usuario y permite utilizar el disco de **forma intuitiva y cómoda** (más cercana a los **hábitos humanos** de organización de la información).
  **Organización** del sistema de archivos utilizando el esquema de **almacenamiento en archivos y la organización en carpetas o directorios**.



**Archivos**: elemento central de los programas de aplicación. Son **estructuras de datos en disco donde se almacena la información** y los programas de un ordenador.

Cada archivo de un sistema tendrá unas **características (atributos)** que lo identifican y le sirven al sistema de archivos y al SO para manejarlo correctamente.
Los atributos pueden variar de un sistema a otro, pero suelen **coincidir** en:

- **Nombre**: identificador principal del archivo para el usuario. Cada SO establece las **reglas** para nombrar a los archivos, longitud, caracteres permitidos, etc.
- **Extensión**: caracteres que se colocan al final del nombre para especificar su tipo de contenido.

*Ejemplo: extensión ".TXT" indica archivo de texto, extensión ".EXE" indica archivo ejecutable.*

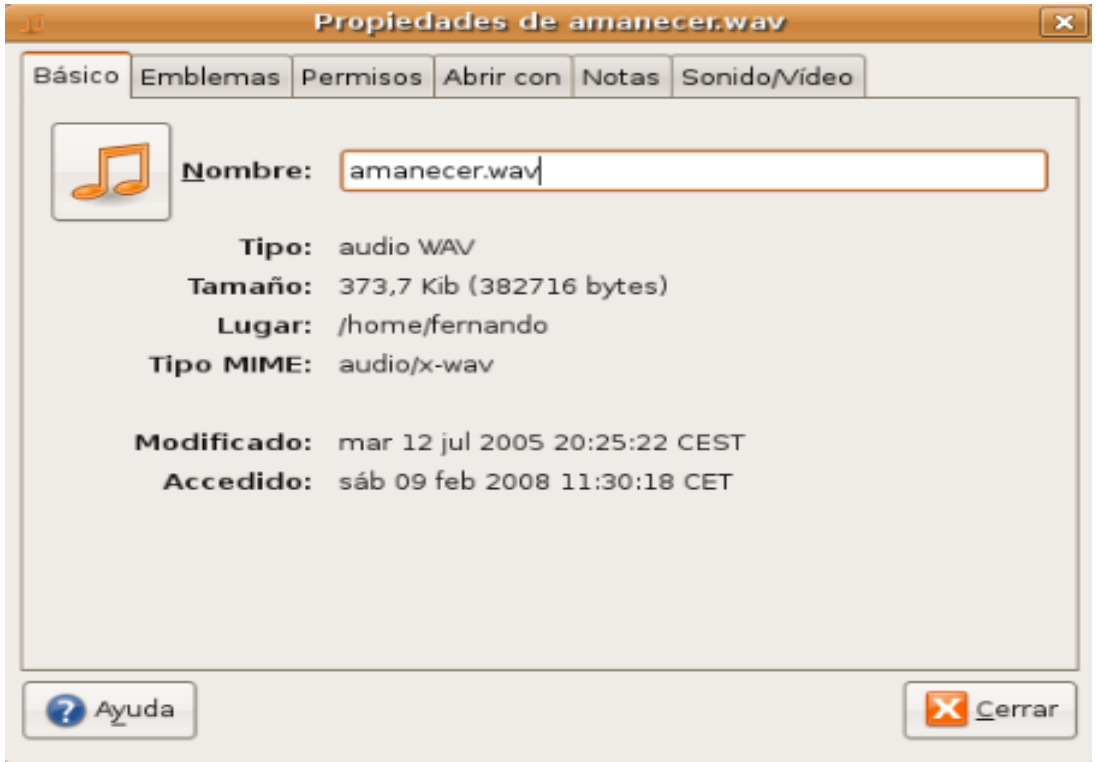
- **Permisos**: el sistema de archivos controla qué **usuarios** están autorizados a utilizar cada archivo y que **operaciones** pueden realizar.

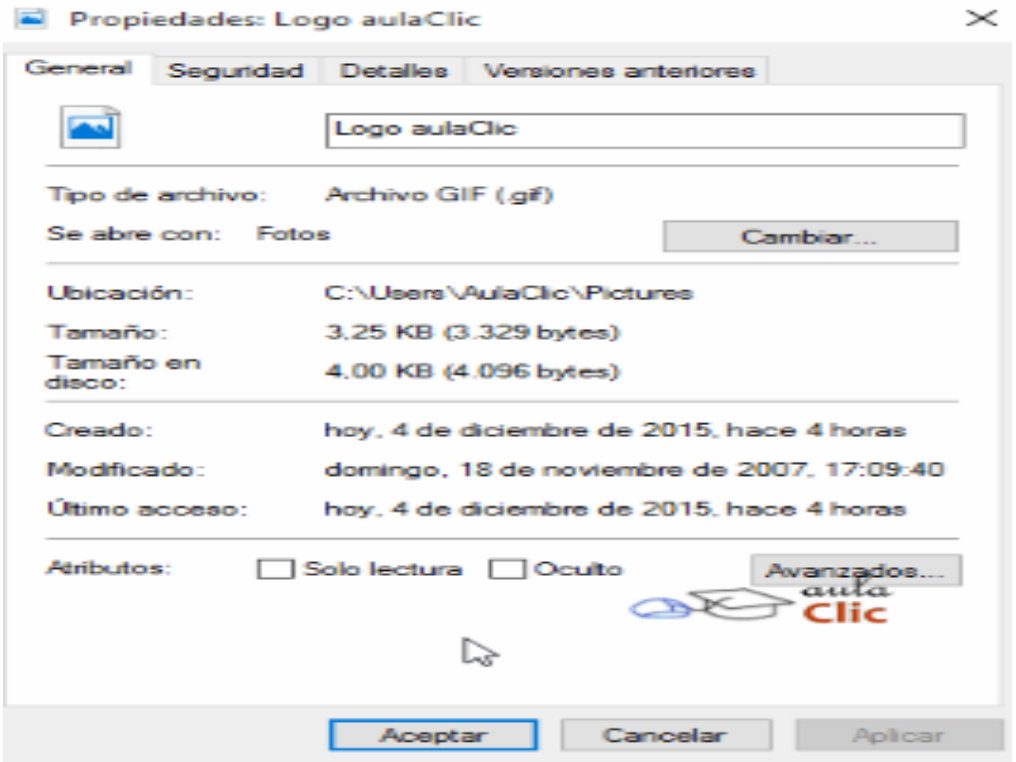
*Ejemplo: permisos de L/E para un usuario.*

- **Creador**: id del **usuario que creo el archivo**.
- **Propietario**: id usuario **propietario actual** del archivo.
- **Fecha de creación**: fecha y hora de **creación** del archivo.

- **Fecha del último acceso**: fecha y hora del **último acceso** al archivo.
- **Fecha de la última modificación**: fecha y hora de la **última modificación** al archivo.
- **Tamaño** actual: número de **bytes que ocupa el archivo** en el disco duro.

**Directorios** (carpetas): archivos especiales que cumplen la función de **almacenar y organizar** en su interior a **archivos** y otros **subdirectorios**. Permiten mantener la organización en el sistema de archivos. En forma de **árbol invertido** que comienza por un **directorio principal** llamado **raíz** y se va **ramificando** en otros directorios que pueden contener archivos y otros directorios.
Respecto a los **atributos** de un directorio, como archivos que son coinciden en los atributos.



## 6.2.- Operaciones soportadas por un sistema de archivos.

Las operaciones **básicas** sobre archivos que la mayoría de los sistemas de archivos soportan son:

- **Crear**: los archivos se crean sin datos y después el usuario o alguna aplicación los llena.
- **Borrar**: si un archivo ya no es necesario debe eliminarse para liberar espacio en disco.
- **Abrir**: antes de utilizar el archivo se debe abrir para que el sistema conozca sus **atributos**, tales como el propietario, fecha de modificación, etc.
- **Cerrar**: tras realizar las operaciones deseadas sobre el archivo, debe cerrarse para **asegurar su integridad y liberar recursos de memoria** que tuviera asignados.
- **Leer**: los datos se leen del archivo, quien hace la llamada (programa) debe especificar la **cantidad** de datos necesarios y proporcionar un **buffer** para colocarlos.
- **Escribir**: los datos se escriben en el archivo. El **tamaño del archivo puede aumentar** si se agregan datos nuevos o no si lo que se hace es actualizar los existentes.
- **Renombrar**: modificar el **atributo nombre** de un archivo ya existente.
- **Crear/Eliminar un enlace:** se utiliza para poder **acceder a un archivo o directorio desde distintos puntos de la organización de directorios** del sistema sin tener que duplicar o copiar el archivo o directorio en cuestión.

## 6.3.- Rutas de acceso.

Los sistemas de archivos necesitan una **forma de determinar la localización** exacta de un archivo o directorio en la estructura del árbol de directorios.

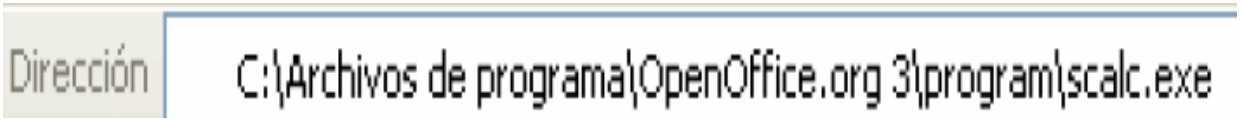
La **ruta de acceso** a un archivo/directorio se indica **nombrando todos los directorios y subdirectorios que tienen que atravesarse** hasta llegar al elemento concreto.

Dependiendo del SO con el que se trabaje cambiará la forma de establecer la ruta de acceso.

*Ejemplo, en Windows se utiliza la barra "\" para separar los directorios y en Linux se utiliza la barra "/".*

Existen **dos tipos** de rutas de acceso:

- **Absoluta**: comienza desde el **directorio raíz y se va descendiendo** en la estructura de directorios **hasta llegar al archivo o directorio** buscado. Se conoce la **ubicación exacta**.




- **Relativa**: concepto de directorio de trabajo o **directorio activo**, aquel donde estamos **situados en un momento dado**. Consiste en escribir la **ruta a partir del directorio activo**, esto se indica con **'..'** que hace referencia a la localización actual donde nos encontramos. N**o se conoce la ubicación exacta**.

Los sistemas de archivos varían de un SO a otro:


- **FAT16** (File Allocation Table)
  - Puede generar particiones de hasta **2Gb**.
  - Cada unidad de disco será **independiente** del resto. Cada disco tendrá su propio sistema de archivos, su propia zona de datos, su tabla FAT independiente, etc.
  - Utiliza **asignación no contigua con encadenamiento**.


- **FAT32**
  - Permite gestionar particiones de hasta **2Tb**.
  - Sólo permite guardar archivos de hasta 4 GB
  - Versiones de Windows son Windows98, Millennium, 2000, XP, NT, etc.).
  - Utiliza **asignación no contigua con encadenamiento**.
  - Anticuado pero robusto. Es tremendamente versátil gracias a su enorme compatibilidad con prácticamente todos los dispositivos y SO.
  - **exFAT** como **una actualización al FAT32** con la intención de acabar con la limitación 4GB.


- **NTFS**
  - Es de **Microsoft**. Usado en las últimas versiones de los SO Windows.
  - Puede gestionar particiones de disco de hasta **2Hexabytes**.
  - Puede considerar más de un disco duro como un único **volumen**.

En **Linux** los sistemas de archivos más utilizados son:


- **ext2**: sistema de archivos nativo Linux.
  - Fragmentación muy baja
  - Lento manejando archivos de gran tamaño.
  - Admite particiones de disco de hasta **4TB** y ficheros de hasta 2GB de tamaño.
  - Proporciona nombres de ficheros largos, de hasta 255 caracteres.
  - Tiene una gran estabilidad.
  - Actualización.

- **ext3**: tiene las propiedades de ext2, pero añade **una bitácora o diario**, que mejora el **rendimiento** y el tiempo de **recuperación** en caso de caída del sistema.
  - **Previsión** de pérdida de datos por fallos del disco o apagones.
  - Es totalmente imposible recuperar datos borrados.
  - Actualización.
  - Fiabilidad y mantenimiento.

*La **bitácora** es un mecanismo que lleva un **registro por cada transacción** que se va a realizar, o que ha sido realizada. Esto permite al sistema de archivos recuperarse fácilmente tras un daño ocasionado, por ejemplo, por cierres del sistema inadecuados.*

- **ext4** (fourth extended filesystem): sistema de archivos **transaccional** (en inglés **journaling**)
  - Anunciado como mejora compatible de ext3.
  - Capaz de trabajar con **volúmenes de gran tamaño**, hasta 1EB y **ficheros** de tamaño de hasta 16TB.
  - **Eficiencia**: menor uso de CPU, mejoras en la velocidad de lectura y escritura.

- **reiserfs**: sistema de ficheros de última generación para Linux.
  - Organiza los ficheros de modo que se agilizan mucho las operaciones con estos.
  - Muchas herramientas (por ejemplo, para recuperar datos) no lo soportan.
  - Utiliza una **bitácora** que provoca que la pérdida de datos sea menos frecuente.
  - Utiliza **asignación no contigua con Indexación**.

- **NFS**: utilizado para compartir recursos entre equipos Linux.
- **SMB**: para compartir recursos entre máquinas Linux y Windows.
- **SWAP**: sistema de ficheros para la partición de **intercambio** de Linux. Partición para cargar los programas y no saturar la memoria RAM cuando se excede su capacidad.

**ESTRUCTURA LÓGICA DE UN DISCO DURO**

Cuando tenemos un disco duro totalmente nuevo, éste no puede ser usado.

**Formateo a bajo nivel** en el cual se definen de forma permanente sobre la superficie de sus platos las **pistas** y **sectores** sobre los que se grabarán los datos que tenga que almacenar. Consiste en colocar **marcas** en la superficie del disco para dividirlo en sectores físicos de 512 bytes e ir numerándolos, para posteriormente poder acceder a ellos cuando sea necesario indicando sus números de cabeza, sector y cilindro.

Hay que crear unas estructuras iniciales para que pueda ser usado: las **Particiones** y el **Master Boot Record (MBR)**.

- Las **particiones** son divisiones lógicas del disco duro, es decir, son partes independientes del disco que nos puede servir para separar la información.

*Ejemplo: es recomendable que los ficheros del SO y los ficheros de los usuarios o copias de seguridad estén separados para poder trabajar individualmente con cada tipo de información.*

- El registro y control de las particiones, se almacena en un **sector especial**, que es el **primero del disco duro:** el Master Boot Record (**MBR**).

**Reglas de particionado basadas en BIOS: MBR (Master Boot Record).**

Las particiones basados en BIOS, se definen y describen en una estructura de datos llamada **Tabla de Particiones** que cada disco se guarda en su **primer sector** y que por cuestiones de espacio sólo tiene **4 registros**, motivo por el cual no puede haber más de 4 particiones primarias.

Por cada partición hay que especificar:

- **principio**, en que sector empieza.
- **final**, en que sector acaba.
- **tamaño**, el número de sectores que ocupa.
- **otros**: identificador de **formato** y el marcador que indica que es la partición **activa**, cuando sea declarada como tal. En cada disco sólo puede haber una partición activa y será la que se utilice para iniciar el sistema.

Cada **disco duro** tiene su tabla de particiones guardada en su MBR para que el SO que lo utilice pueda usarlas con normalidad.

Existen **tres** tipos de particiones:

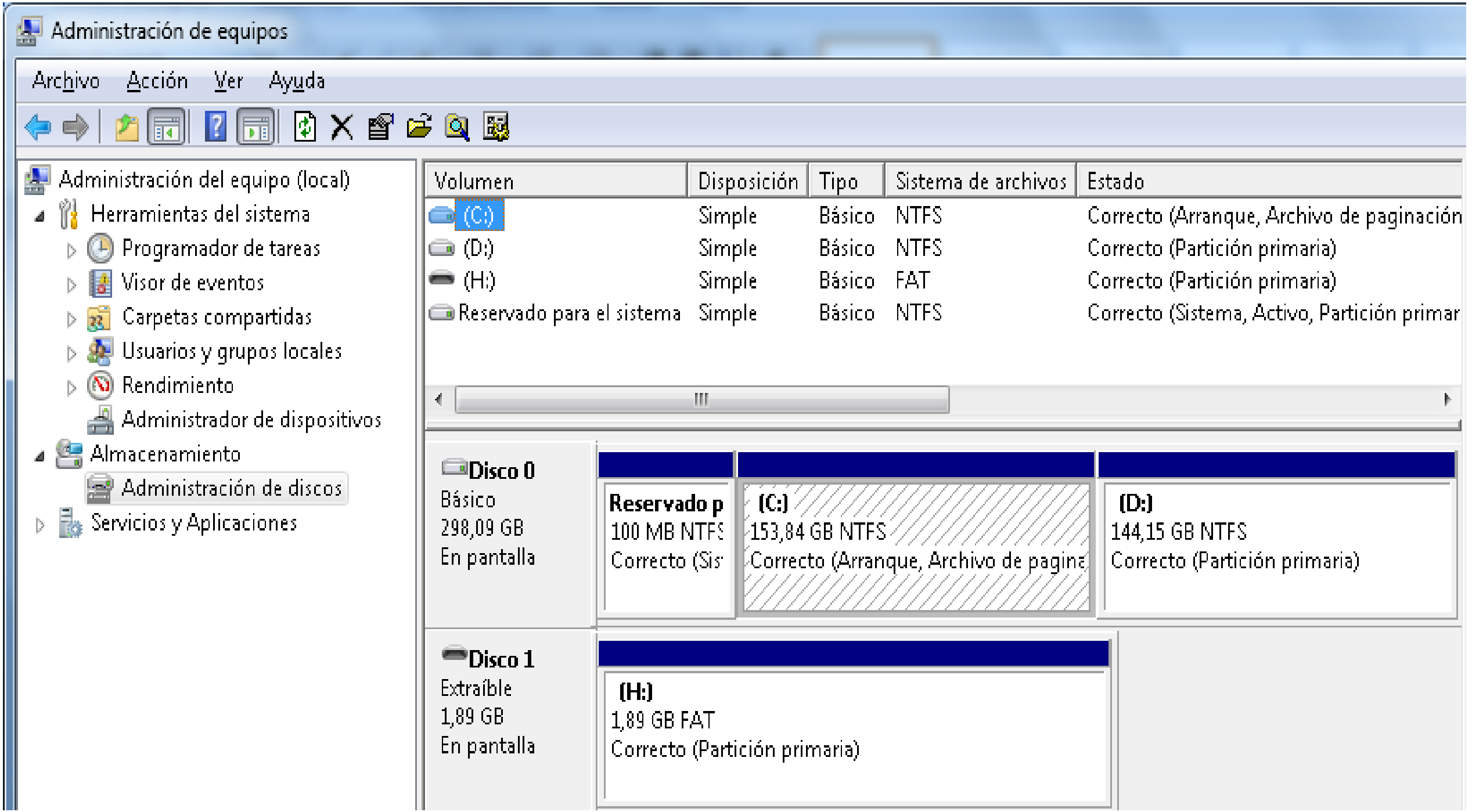
- **Primarias**: este tipo de particiones son divisiones directas del disco duro. El problema es que solo pueden existir un máximo de **4 particiones primarias** gestionadas en el MBR.

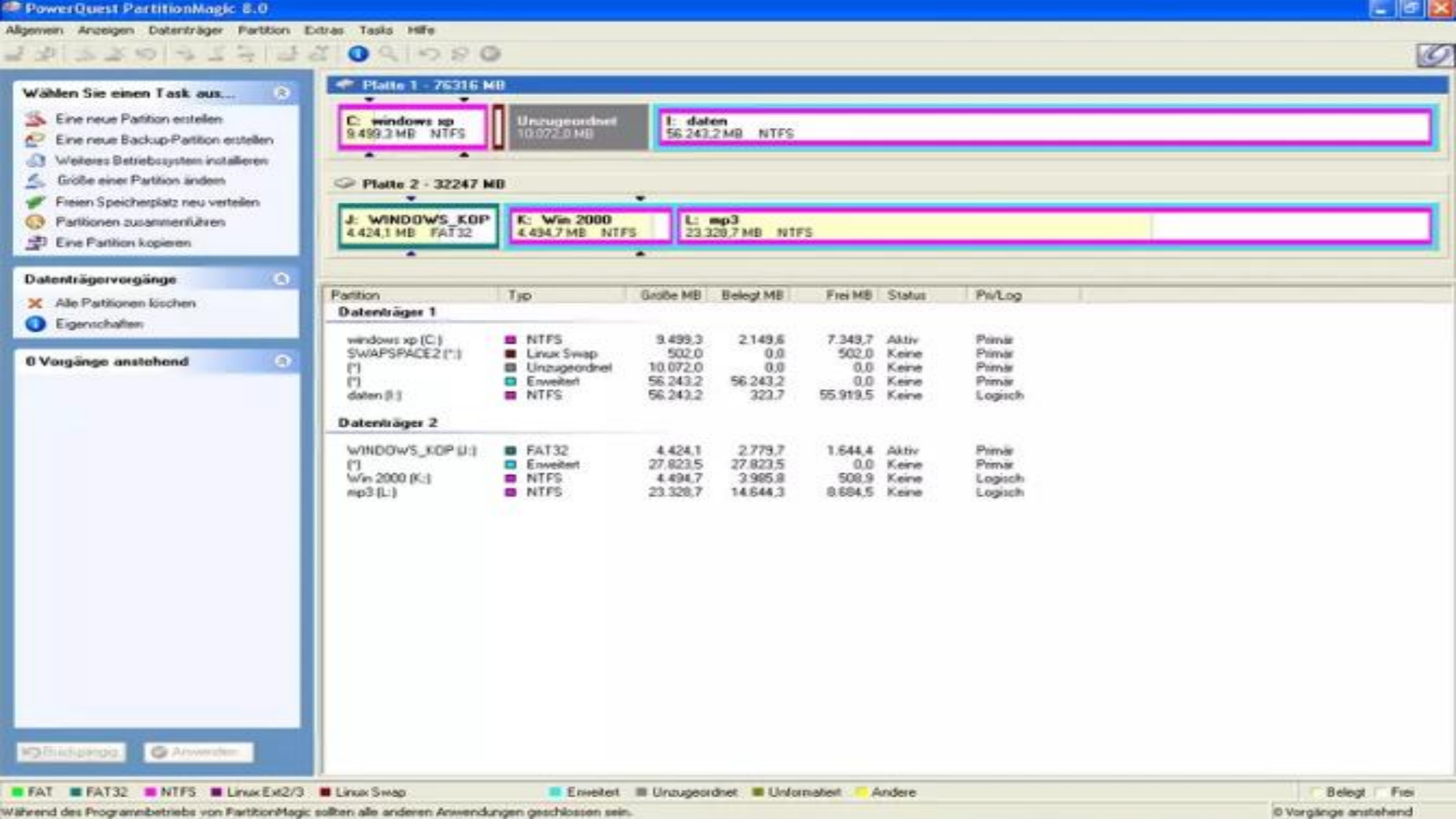
- **Extendida o secundaria**: **no** puede ser usada para **guardar información**. Es una división del disco que a su vez puede ser particionada hasta un máximo de 23 particiones lógicas. Sólo puede haber **una** partición de este tipo.

- **Lógicas**: particiones dentro de la partición extendida. Pueden ser usadas como las primarias, tanto para instalar SO como para guardar información.

La estructura lógica de un disco puede manipularse con programas de gestión de particiones:

- **Windows** incorpora el Administrador de Discos.
- **GParted** o **Partition Magic** son software comercial de manipulación de particiones.


Administración de equipos
Archivo Acción Ver Ayuda
Administración del equipo (local)
Herramientas del sistema
Programador de tareas
Visor de eventos
Carpetas compartidas
Usuarios y grupos locales
Rendimiento
Administrador de dispositivos
Almacenamiento
Administración de discos
Servicios y Aplicaciones
Volumen Disposición Tipo Sistema de archivos Estado
(C:) Simple Básico NTFS Correcto (Arranque, Archivo de paginación
(D:) Simple Básico NTFS Correcto (Partición primaria)
(H:) Simple Básico FAT Correcto (Partición primaria)
Reservado para el sistema Simple Básico NTFS Correcto (Sistema, Activo, Partición primar
Disco 0
Básico
298,09 GB
En pantalla
Reservado p
100 MB NTFS
Correcto (Sis
(C:)
153,84 GB NTFS
Correcto (Arranque, Archivo de pagina
(D:)
144,15 GB NTFS
Correcto (Partición primaria)
Disco 1
Extraíble
1,89 GB
En pantalla
(H:)
1,89 GB FAT
Correcto (Partición primaria)


PowerQuest PartitionMagic 8.0
Allgemein Anzeigen Datenträger Partition Extras Tasks Hilfe
Wählen Sie einen Task aus...
Eine neue Partition erstellen
Eine neue Backup-Partition erstellen
Weiteres Betriebssystem installieren
Größe einer Partition ändern
Freien Speicherplatz neu verteilen
Partitionen zusammenführen
Eine Partition kopieren
Datenträgervorgänge
Alle Partitionen löschen
Eigenschaften
0 Vorgänge anstehend
Rückgängig
Anwenden
Platte 1 - 76316 MB
C: windows xp
9.499,3 MB NTFS
Unzugeordnet
10.072,0 MB
I: daten
56.243,2 MB NTFS
Platte 2 - 32247 MB
J: WINDOWS_KOP
4.424,1 MB FAT32
K: Win 2000
4.494,7 MB NTFS
L: mp3
23.328,7 MB NTFS
Partition Typ Größe MB Belegt MB Frei MB Status Pri/Log
Datenträger 1
windows xp (C:) NTFS 9.499,3 2.149,6 7.349,7 Aktiv Primär
SWAPSPACE2 (*:) Linux Swap 502,0 0,0 502,0 Keine Primär
(*:) Unzugeordnet 10.072,0 0,0 0,0 Keine Primär
(*:) Erweitert 56.243,2 56.243,2 0,0 Keine Primär
daten (I:) NTFS 56.243,2 323,7 55.919,5 Keine Logisch
Datenträger 2
WINDOWS_KOP (J:) FAT32 4.424,1 2.779,7 1.644,4 Aktiv Primär
(*:) Erweitert 27.823,5 27.823,5 0,0 Keine Primär
Win 2000 (K:) NTFS 4.494,7 3.985,8 508,9 Keine Logisch
mp3 (L:) NTFS 23.328,7 14.644,3 8.684,5 Keine Logisch
FAT FAT32 NTFS Linux Ext2/3 Linux Swap Erweitert Unzugeordnet Unformatiert Andere
Belegt Frei
Während des Programmbetriebs von PartitionMagic sollten alle anderen Anwendungen geschlossen sein.
0 Vorgänge anstehend

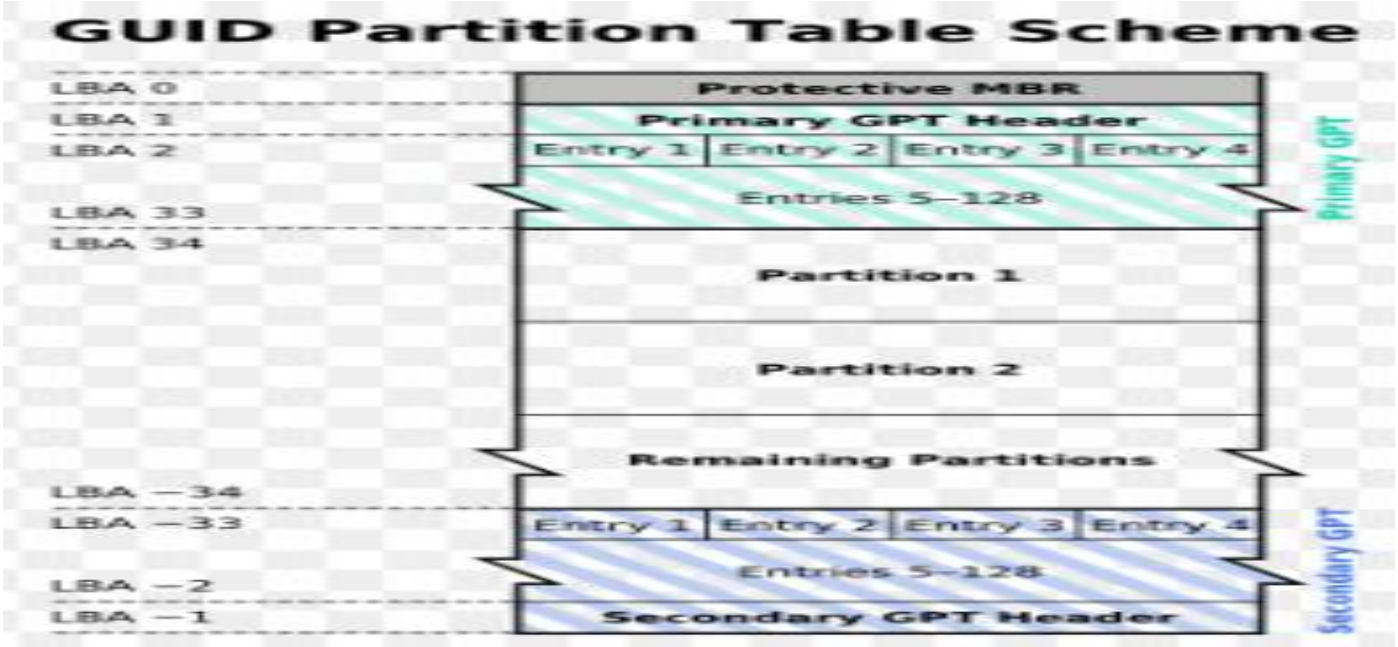
**Reglas de particionado basadas en el estándar UEFI** (Unified Extensible Firmware Interface)**: GPT (GUID Partition Table).**

- Sustituir al estándar **BIOS**.
- Los discos utilizan una **tabla de particiones GPT** en la que se pueden declarar hasta **128** particiones primarias, lo que hace innecesario el uso de particiones extendidas y lógicas.
- Mantiene una **segunda copia redundante** de la tabla de particiones al final del disco duro.
- Usa el método de direccionamiento **LBA** (Logical Block Addressing) para **especificar la localización de los bloques** que lo forman.

Se **estructuran** como sigue:

- **LBA 0**: en el primer bloque de cada disco, se sitúa un **MBR "heredado",** que se mantiene por **compatibilidad** con el anterior esquema BIOS y como protección contra antiguas herramientas software de disco duro que no reconocen el particionado GPT y no saben como acceder correctamente a sus particiones.

- **LBA 1:** cabecera en la que se definen los bloques de disco que están disponibles para ser utilizados por los SO para su uso normal.
  - Se definen las **dimensiones de la tabla de particiones** (número máximo de particiones que podrá tener el disco duro, 128) y el **tamaño** en bytes para cada una de las entradas de partición (128bytes de longitud).
  - Se guarda el **tamaño** y el **GUID** del **disco** (Globally Unique Identifier), y el **emplazamiento** de las cabeceras de partición y tablas de particiones, de ella misma (LBA1), y de la secundaria (último sector del disco).
  - Finalmente contiene una suma de **comprobación CRC32** para la cabecera y para la tabla de particiones, que se verifica por los procesos UEFI durante el **arranque**.



- **LBA 2 a LBA 33:** entradas de partición.

Se utilizan para registrar la **tabla de particiones** que están compuestas de:

- Primeros 16bytes designan el **tipo de partición GUID**. Números pseudo aleatorios que identifican las particiones.

  *Ejemplo: GUID para una partición EFI {28732AC1-1FF8-D211-BA4B-00A0C93EC93B}*
- Los siguientes 16bytes que contienen otro **GUID único** para la partición.
- Los **bloques LBA de inicio de la partición y de su final**, que se registran codificados como enteros de 64 bits.
- Espacio para guardar el **nombre** (hasta 36 caracteres ) de las particiones y otros **atributos**.

Para discos con capacidades superiores a esos 2 Terabytes, que probablemente será necesario dividir en más de cuatro particiones, empieza a ser más que recomendable utilizar el sistema de particionado GPT.

## ESTRUCTURA LÓGICA DE UNA PARTICIÓN

Para poder usar una partición se necesita **crear y saber interpretar una estructura** que se aplica a dicha partición. Estas estructuras se crean en el proceso de **formateo** de **alto nivel** de partición.

El proceso de formato lo que hace es **crear las estructuras necesarias en la partición** (excepto en la partición extendida) para que un SO pueda guardar y recuperar la información en la memoria secundaria. Estas estructuras son el **sistema de ficheros**.

Mientras se formatea se hace una **revisión de la superficie magnética** del disco, y si se detectan **sectores defectuosos** se marcan como erróneos para no usarlos.

Tener en cuenta que al formatear una partición usada, se **perderá** todo su contenido

Al formatear una partición se crean las siguientes **zonas**:

- **Sector de Arranque**: contiene las instrucciones necesarias para iniciar una partición.

- **Sistema de Ficheros**: parte del SO necesaria para poder acceder a la información de los ficheros. Cada sistema de ficheros establece unas propiedades concretas necesarias por el SO para el manejo de los ficheros y directorios. Esta parte hace como **índice**.

*Ejemplo: **índice** de un libro. La información relevante son los temas o capítulos del libro, si un usuario necesita buscar un capítulo concreto existe un índice al principio donde se ponen los títulos y las páginas donde se encuentra la información necesaria.*

- Zona de datos o **cluster**: parte donde realmente se guarda el contenido de un fichero.