

SISTEMAS OPERATIVOS MONOUSUARIOS

UNIDAD 2

FUNCIONAMIENTO GENERAL DE LOS SISTEMAS OPERATIVOS.

ÍNDICE

1.	Introducción	2
2.	Evolución histórica de los S.O.	3
3.	Clasificación de los Sistemas Operativos	5
3.1.	Según el número de usuarios	5
3.2.	Según el número de procesos	5
3.3.	Según el número de procesadores	6
3.4.	Según el tiempo de respuesta	6
4.	Estructura de los Sistemas Operativos	7
5.	Funciones del Sistema Operativo	8
6.	Gestión de procesos y del procesador	9
6.1.	Estados de un proceso	10
6.2.	Hilos de ejecución	11
6.3.	Gestión de procesos en Linux y Windows	12
6.4.	Planificación de procesos en la multiprogramación	14
6.4.1.	Algoritmo de rueda, de petición circular, cooperativo o Round Robin :	14
6.4.2.	Algoritmo por intervalos de espera	15
6.4.3.	Algoritmo FIFO o FCFS (Primero en entrar, primero en salir)	15
6.4.4.	Algoritmo SJF (El más corto primero)	15
6.4.5.	Algoritmo SRNT (El menor tiempo restante a continuación)	15
7.	Gestión de memoria	16
7.1.	Particiones estáticas	16
7.2.	Particiones dinámicas	17
7.3.	Paginación	18
7.4.	Segmentación	18
7.5.	Memoria virtual	19
7.6.	Swapping	20
8.	Gestión de archivos	21
8.1.	Sistemas de archivos FAT	23
8.2.	Sistemas de archivos NTFS	24
8.3.	Sistemas de archivos ext para Linux	25
8.4.	Sistemas de archivos HFS para MAC	25
8.5.	Otros sistemas de archivos	25
9.	Gestión de entrada y salida.	27
9.1.	Interrupciones	28
9.2.	DMA o Acceso Directo a Memoria	29
9.3.	Gestión de dispositivos de entrada y salida lentos	29
10.	Bibliografía y Webgrafía	31
11.	ANEXO 1. Algoritmo Round Robin	32
11.1.	Round Robin sin E/S	32
11.2.	Round Robin con E/S	32
12.	ANEXO 2. Algoritmo por intervalos de espera	33
13.	ANEXO 3. Algoritmo FIFO ó FCFS	34
14.	ANEXO 4. Algoritmo SFJ	35
15.	ANEXO 5. Algoritmo SRNT	36
16.	ANEXO 6. Ejemplo de paginación	37
17.	ANEXO 7. Ejemplo de Segmentación	38

1. Introducción

El S.O.¹ es el software básico del ordenador. Este software gestiona todos los recursos del sistema (recursos hardware) y proporciona la base para la creación y ejecución del software de aplicación (programas y aplicaciones propiamente dichas).

Un Sistema Operativo es un programa (o conjunto de programas) de control que tiene por objeto facilitar el uso del ordenador y conseguir que éste se utilice eficientemente. Es un programa de control, ya que se encarga de gestionar y asignar los recursos hardware a los usuarios. Controla los programas de estos y los dispositivos de entrada y/o salida.

El S.O. es el que realiza “el trabajo sucio” dentro del equipo, ya que oculta al usuario la verdad acerca del hardware. Gracias a una interfaz² sencilla proporciona al usuario una comunicación directa sin que éste tenga que preocuparse de la gestión de memoria, del procesador o de cualquier otro recurso o componente hardware o software.

Supongamos que estamos trabajando con un sistema operativo que permite la ejecución de más de un programa a la vez. Supongamos, entonces, que se están ejecutando simultáneamente varios programas de un mismo ordenador y que en un momento determinado las diferentes aplicaciones que se están ejecutando necesitan hacer uso de la impresora para imprimir resultados. El caos podría ser tremendo si no hubiese algo o alguien que controlase una salida ordenada de la información por la impresora. Es entonces cuando interviene el S.O. para organizar la salida de la información determinando qué aplicación tiene prioridad, qué impresora se va a utilizar, qué tipo de papel es el deseado, etc...

Pero esto se complica aún más cuando varios usuarios quieren utilizar el sistema simultáneamente, ya que, para ellos, el uso del ordenador tiene que ser transparente; es decir, es como si cada usuario utilizase el ordenador de forma autónoma, independientemente del resto de los usuarios. Pues bien, esta gestión la realiza el S.O.

En resumen, el S.O. hace de intermediario y controlador entre la parte física del ordenador, el software que se utiliza, y el usuario para gestionar y administrar sus recursos.

La interfaz es la forma en que interactúa el usuario y el sistema operativo. Es un conjunto de servicios que ofrece el S.O. y que pueden invocarse, esos servicios se denominan **llamadas al sistema**; la forma de invocarlos es mediante programas, desde el intérprete de órdenes o desde el gestor de ventanas.

En cada S.O. las llamadas son diferentes aunque su funcionalidad sea similar. Ejemplo de llamada sería abrir una ventana, mover un archivo, etc.

¹ S.O. : Sistema Operativo

² Interfaz: medio de comunicación entre usuario y equipo.

2. Evolución histórica de los S.O.

Los S.O. han estado siempre relacionados con las arquitecturas de los ordenadores. Históricamente podemos hablar de cuatro generaciones de ordenadores cuyas características quedan definidas por los componentes hardware en los sistemas informáticos.

De forma genérica podemos hablar de varias generaciones de sistemas operativos:

➤ **Primera generación (1945-1955) :**

Utilizaban las válvulas de vacío (antiguas resistencias electrónicas) para la construcción de las computadoras. Estas máquinas eran programadas en lenguaje máquina puro. Eran de gran tamaño, consumían mucha energía y eran muy lentas en el tratamiento de las operaciones, que se reducían a cálculos matemáticos. A principios de los años cincuenta, para introducir datos al ordenador se utilizaban las tarjetas perforadas. Anteriormente, toda la información que se debía procesar se introducía de forma manual.

➤ **Segunda generación (1955-1965) :**

Se integraron transistores dentro la arquitectura de las computadoras. Por otro lado, cada persona del equipo informático que hacía funcionar la máquina se encargaba de realizar una labor concreta.

En esta generación aparece lo que se denomina procesamiento por lotes, que consiste en que los datos son introducidos al ordenador que los va a procesar por otro pequeño ordenador o computador. Este proceso constaba de tres labores fundamentales:

1. Introducir los datos que se iban a procesar en un soporte magnético o no. Los primeros soportes de información fueron las tarjetas perforadas o la cinta perforada. La introducción de datos se realizaba en un medio físico distinto de la computadora.
2. Llevar el soporte cargado con los datos a la computadora para que los procesara. Procesada la información, se almacenaba en otro soporte diferente.
3. Llevar el soporte donde están los resultados a otro dispositivo físico distinto a la computadora para realizar la generación de cálculos.

➤ **Tercera generación (1965-1980) :**

Aparecen los circuitos integrados. Se reduce considerablemente el tamaño y consumo de energía de los ordenadores. Por otro lado, son más baratos y rápidos.

Es de destacar el IBM 360 como máquina capaz de realizar cualquier tipo de cálculo.

También aparece el concepto de multiprogramación; es decir, permitir que una máquina esté realizando varios procesos a la vez, y aparecen las técnicas de spool, gracias a las cuales se almacenan trabajos en colas de espera. De esta forma, se podían ejecutar varios programas, ya que cuando finalizaba uno, el S.O. podía cargar otro en memoria para ejecutarlo.

Así, para gestionar estos grandes ordenadores, era necesaria la construcción de un software básico que controlase y relacionase los diferentes componentes del ordenador. Este S.O. era complicado, con gran cantidad de errores y con muy pocos recursos.

➤ **Cuarta generación (1980 – Hoy) :**

Destacada esta generación por la aparición de los ordenadores personales. Se utilizan complejas técnicas de integración de componentes electrónicos. Aparecen las memorias de semiconductores, que son mucha más pequeñas, mucho más rápidas y de mayor capacidad.

Aparecen S.O. mucho más fáciles de usar; es decir, con mayores posibilidades. Para ello, se crean interfaces sencillas de comunicación entre usuario y máquina.

Surgen S.O. como el MSDOS, los N.O.S.³, Windows, en sus diferentes versiones y LINUX, que permiten un diálogo con el ordenador basado en entornos gráficos. Estos S.O. actuales son de elevada potencia, sobre todo en la gestión de hardware y en la utilización y distribución

³ N.O.S. : Siglas en ingles de Network Operative System, Sistemas Operativos en Red

3. Clasificación de los Sistemas Operativos

Para hacer una clasificación de los S.O. hay que tener en cuenta una serie de parámetros:

- Número de usuarios
- Número de procesos
- Número de procesadores
- Tiempo de respuesta.

3.1. Según el número de usuarios

- **Monousuarios:** Sólo un usuario trabaja con un ordenador. En este sistema todos los dispositivos de hardware están a disposición de dicho usuario y no pueden ser utilizados por otros hasta que éste no finalice su sesión. Ejemplos de estos sistemas son: MSDOS y los sistemas operativos Windows 3.1, Me, XP Home y Profesional, Vista y Windows 7.
- **Multiusuarios:** En este sistema, varios usuarios pueden utilizar simultáneamente los recursos del sistema. Pueden compartir, sobre todo, los dispositivos externos de almacenamiento y los periféricos de salida, fundamentalmente impresoras, por ejemplo, los trabajos enviados por varios usuarios se sitúan en colas de espera hasta que les llegue su turno. Estas colas se denominan **spool de impresión**. Ejemplos de sistemas multiusuarios son UNÍX, Novell, Windows NT Server, Windows 2000 Server, VMS, MVS (Grandes equipos de IBM)

3.2. Según el número de procesos

Esta clasificación se hace atendiendo al número de programas o procesos que se pueden realizar o ejecutar simultáneamente el ordenador.

- **Monoprogramación o monotarea:** En este caso, el sistema solamente puede ejecutar un programa a la vez. De esta forma, los recursos del sistema estarán dedicados al programa hasta que finalice su ejecución.
Esto no impide que el sistema pueda ser multiusuario; es decir, varios usuarios pueden intentar ejecutar sus programas en el mismo ordenador, pero de forma sucesiva. Para ello, se tienen que establecer las correspondientes colas o prioridades en la ejecución de los trabajos.
En este sistema, la atención del procesador estará dedicada a un solo programa hasta que finalice. Ejemplo claro lo tenemos en MS-DOS, Windows 95 y 98.
- **Multiprogramación o multitarea:** Con estos sistemas se pueden ejecutar varios programas o procesos concurrentemente. Para ello la CPU compartirá el tiempo de uso del procesador entre los diferentes programas que se van a ejecutar.
Así, todos los procesos tardarán individualmente más tiempo en ejecutarse; pero, comparándolo con la monoprogramación, el tiempo medio de espera será mucho menor. Ejemplos claros de estos S.O. son los vistos en la clasificación de multiusuario y los sistemas Windows XP, Vista y Windows 7 y 8.

3.3. Según el número de procesadores

- **Monoprocesador:** En este caso, el ordenador consta de un único procesador. Todos los trabajos pasarán por él. El ordenador que tenga este S.O. puede ser mono o multiusuario y mono o multitarea.
- **Multiprocesador:** El ordenador cuenta con varios procesadores. Estos pueden actuar de dos formas distintas. Existen ordenadores que irán saturando de trabajo a sus procesadores poco a poco. Con la primera tarea utilizará el primer procesador; si entra otra tarea, se utilizará lo que reste de potencia del primer procesador y lo necesario del segundo. Los demás procesadores se irán utilizando de forma sucesiva. De esta forma pueden quedar procesadores inactivos. Por otro lado, hay sistemas que utilizarán la totalidad de los procesadores que tienen para realizar todas las tareas, es decir, la saturación del procesador sólo se producirá cuando el sistema esté funcionando al cien por cien. Cada programa utilizará parte de todos los procesadores. Si llega otro nuevo programa para ser ejecutado, se utilizarán también todos los procesadores, y así hasta su total utilización. De esta forma trabajarán todos, pero es evidente que lo harán a bajo rendimiento. Esta técnica es usada por los S.O. Windows NT, 2000 y muchas versiones de UNIX. Es lo que se denomina multiproceso simétrico.

Hay sistemas que dedican un procesador a cada programa distinto. Con esto, se consigue que programas sencillos se ejecuten en poco tiempo, pero los complejos tardarán mucho tiempo en realizarse. De esta forma, todos los procesadores trabajarán a un rendimiento medio.

3.4. Según el tiempo de respuesta

- **Tiempo real:** La respuesta es inmediata o casi inmediata, tras lanzar un proceso.
- **Tiempo compartido:** Cada proceso utilizará fracciones de tiempo de ejecución de la CPU hasta que finalice. En este caso, parece que el usuario dedica la CPU exclusivamente para él; pero esto no es cierto, ya que, aunque el usuario no lo perciba, la CPU está dedicada a varios procesos a la vez. Todos los S.O. multiusuario ofrecen a los usuarios un tiempo de respuesta compartido.

4. Estructura de los Sistemas Operativos

Cada S.O. tiene una estructura propia que debe ser transparente al usuario. Algunas de las estructuras más utilizadas a lo largo de la historia de los sistemas operativos son las siguientes:

- **Sistemas monolíticos:** El S.O. está formado por un conjunto de procedimientos de forma que cada uno puede llamar a los demás cuando los necesite. Todas las funciones que realiza se llevan a cabo con un solo programa que se ejecuta en modo Kernel. Ejemplo de estos sistemas son UNIX y Linux.
- **Sistemas de capas:** Consiste en organizar el S.O. mediante una jerarquía de capas, cada una de las cuales tiene unas funciones asignadas.
- **Microkernels:** La idea de esta estructura es dividir el S.O. en módulos pequeños, ejecutándose sólo uno en modo Kernel, de esta forma un error en un módulo no afecta a todo el sistema, además tienen la ventaja de ser fáciles de mantener debido a su estructura modular. Estos módulos se ejecutan mediante **llamadas al sistema** también conocidas como **instrucciones virtuales** para diferenciarlas de las verdaderas instrucciones que son las que ejecuta la CPU. Ejemplo de esta estructura es el S.O. Symbian.
- **Máquinas Virtuales ó Máquinas Operativas:** Se trata de realizar copias exactas del hardware que tiene la máquina real incluyendo modo kernel y modo usuario, la entrada y salida, etc. Cada máquina virtual es idéntica al verdadero hardware y cada una puede ejecutar un S.O. distinto.
- **Exokernels:** Consiste en dividir los recursos y asignar a cada usuario una parte de ellos. Ejemplos poco conocidos son los S.O. Aegis y XOK.

5. Funciones del Sistema Operativo

Podemos decir que un S.O. debe encargarse de gestionar los siguientes recursos: el procesador, la memoria, la entrada y salida y los archivos del sistema.

- **Gestión de procesos:** El proceso principal se define como programa en ejecución. El S.O. se encarga de crear y destruir procesos, suspender y reanudarlos y sincronizar y comunicar procesos.
- **Gestión de memoria:** La memoria principal se encarga de almacenar procesos e información de procesos que se están ejecutando en el procesador. El S.O. asigna y libera la memoria, decide cuánta memoria se asigna a un proceso y controla las partes de la memoria que están en uso.
- **Gestión de archivos:** El S.O. gestiona los archivos mediante el sistema de archivos, definiéndose como el conjunto de normas y procedimientos para almacenar información en los dispositivos de almacenamiento.
- **Gestión de entrada y salida:** El S.O. controla los dispositivos de E/S, se encarga de capturar interrupciones, enviar y manejar datos de y hacia la memoria que se recogen o envían desde los dispositivos.

6. Gestión de procesos y del procesador

La gestión del procesador va íntimamente vinculada a la gestión de procesos. Podemos decir que un proceso es un programa que se encuentra en la memoria interna en algún punto de su ejecución. De otra forma, podríamos decir que es una abstracción de un programa en ejecución.

Los programas son los archivos almacenados en los medios de almacenamiento del ordenador. Cuando un programa es ejecutado por el sistema o algún usuario, esto crea una entidad de ese programa en la memoria llamada proceso. Por tanto, cada vez que un programa es ejecutado crea una nueva entidad de él y por tanto podemos tener varios procesos en marcha del mismo programa.

Cuando un programa se ejecuta, el S.O. crea un proceso y le asigna a este un **espacio de direcciones** y lo añade a una tabla llamada **tabla de procesos**.

En ese espacio de direcciones asignadas, el proceso guardará las instrucciones que debe ejecutar y los datos necesarios para esa ejecución. Además, cada proceso tendrá asignada una pila que llevará controlará el número de llamadas a procedimientos.

La tabla de procesos contiene información de cada uno de los procesos que se están ejecutando. Cada una de las entradas (una por proceso) contiene los siguientes campos de información.

- **PID:** Son las siglas en ingles de Identificador de Proceso (Process ID). Es un número único que se asigna a un proceso cada vez que se carga en memoria.
- **Estado del proceso:** Los tres estados que guarda este registro son preparado, en ejecución o bloqueado.
- **Prioridad:** el S.O. puede asignar un nivel de prioridad a cada proceso. Con ello podemos dar “una importancia” mayor o menor a cada uno de ellos para su ejecución.
- **Dirección de la memoria:** Es un puntero a la posición de la memoria donde se encuentra el proceso.
- **Directorio de trabajo:** Directorio donde el usuario o sistema almacena el programa en el disco.
- **Tiempo utilizado por el procesador:** Es el tiempo que el proceso se ha estado ejecutando en el procesador.

6.1. Estados de un proceso

Desde que un programa se encuentra en el dispositivo de almacenamiento hasta que finaliza la ejecución del proceso que le corresponde, puede pasar por numerosos estados.

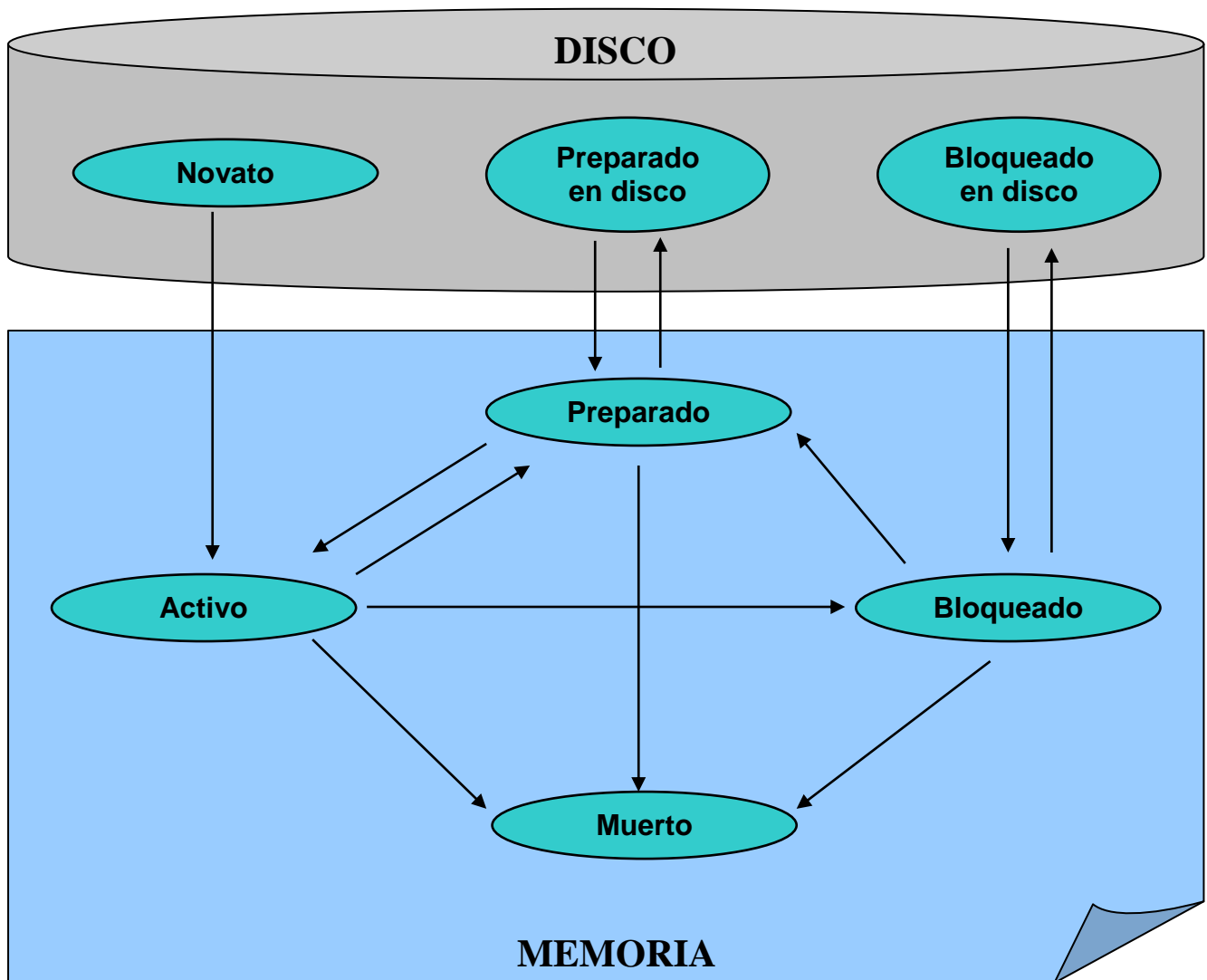


Figura 1. Posibles estados de un proceso

Un proceso, desde el punto de vista de su ejecución, puede estar en una diversidad de situaciones o estados. Aquí resumiremos los diferentes estados.

- Un **proceso novato** es aquel que no ha iniciado su ejecución.
- Un proceso está en **estado preparado**, *listo o ejecutable* cuando se encuentra en memoria principal, sin operaciones de Entrada / Salida pendientes y apto para entrar (o continuar) en ejecución.
- El *estado de ejecución* o **activo** corresponde al proceso que en ese momento está siendo atendido por la CPU. En un sistema de tiempo compartido, sale del estado de ejecución cuando pasa a **estado de bloqueado**.

- Del estado de bloqueado puede pasar al estado preparado cuando acaba la operación de Entrada / Salida pendiente o se le asigna el recurso esperado.
- En un sistema en que se gestiona la memoria con intercambiabilidad y con más procesos en ejecución concurrente de los que admite la memoria principal, un proceso puede ser trasvasado a disco pasando al **estado bloqueado en disco** o *bloqueado intercambiado*.
- Si la carga de procesos es grande, incluso los procesos preparados pueden trasvasarse a disco, pasando así al **estado de preparado intercambiado** o **preparado en disco**.
- Cuando finaliza la ejecución de un proceso o se detecta un error grave en él, el proceso pasa a **estado de muerto**, liberando el S.O. la zona de memoria ocupada.

Además, los procesos más importantes, mientras avanzan en su ejecución, pueden ser obligados a residir en memoria principal. Estos procesos se dice que son **residentes** o *no intercambiables*.

Tal y como se muestra en la Figura 1, y en la explicación que la sigue, el proceso a lo largo de su vida pasa por varios estados. Cada paso de un estado a otro se denomina transición.

6.2. Hilos de ejecución

Un proceso puede a su vez crear subprocesos o hilos de ejecución que le permitirán realizar varias tareas más pequeñas o tareas recursivas de forma simultánea como pueden ser lecturas de disco, esperar un clic de ratón, actualizar un valor en la pantalla, etc. Realizando una ejecución de un proceso mediante el uso de hilos o subprocesos, si alguno de estos hilos queda bloqueado no bloqueará la ejecución del proceso principal.

Imaginemos un ejemplo, necesitamos un programa que nos muestre una pantalla en la que debemos introducir un usuario y una contraseña, comprobar que pertenecen a nuestros usuarios en una base de datos y para ello tendrá un tiempo limitado para lo cual se pondría un temporizador en pantalla.

Es evidente que cuando el proceso entra en ejecución, el temporizador debería comenzar a funcionar, el contador de tiempo de la pantalla debería ir actualizándose y cuando el usuario entre los valores deberá comprobar si son correctos en la base de datos. Cada una de estas tres funciones podría realizarse de forma independiente, de esta forma podría ocurrir un error en el hilo de ejecución del contador de la pantalla pero el proceso seguiría funcionando correctamente, de la misma forma puede interesarnos que no se bloqueara si el contador de tiempo dejara de funcionar, etc.

6.3. Gestión de procesos en Linux y Windows

GESTIONAR PROCESOS EN UBUNTU

Para conocer los procesos que tenemos ejecutándose en un S.O. basado en Linux, como puede ser Ubuntu, tenemos multitud de herramientas en función de la distribución elegida.

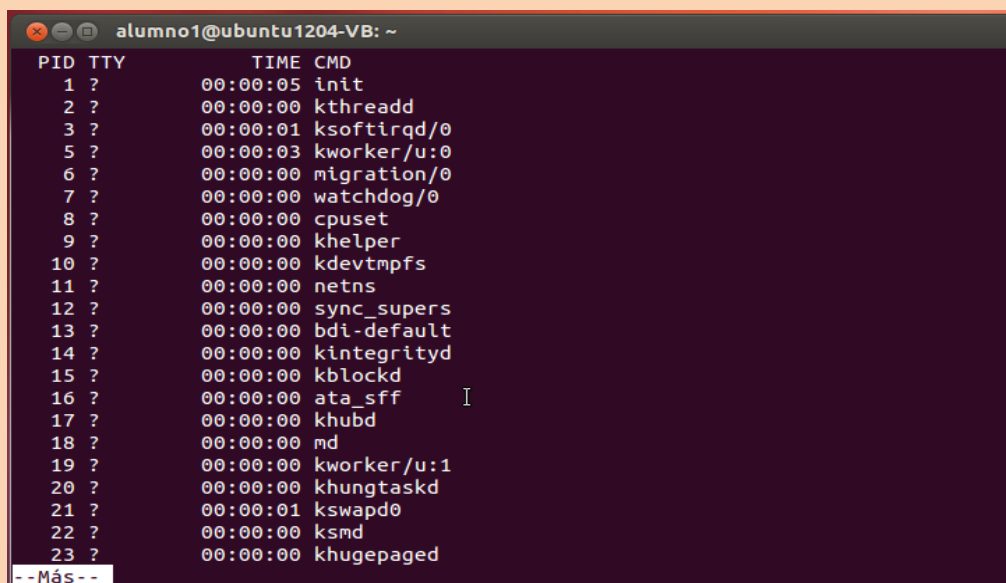
Nosotros vamos a elegir la forma genérica que es la que se realiza utilizando la línea de comandos de Ubuntu. Para ello, ejecutamos una ventana de **Terminal** y una vez dentro de él, ejecutamos la orden **ps**. Este nos mostrará la siguiente información en la ventana:



```
alumno1@ubuntu1204-VB: ~  
ps  
alumno1@ubuntu1204-VB:~$ ps  
  PID TTY          TIME CMD  
 4066 pts/2    00:00:02 bash  
 4120 pts/2    00:00:00 ps  
alumno1@ubuntu1204-VB:~$
```

Esta forma de ejecutar el comando es la más básica y nos mostrará tres columnas con el PID, el terminal asociado al proceso, el tiempo acumulado de CPU y el nombre del archivo ejecutable correspondiente al proceso.

Para ver la información de todos los procesos que actualmente se están ejecutando en el sistema, habría que ejecutar el comando con el modificador **-A** de la siguiente forma: **ps -A**. El resultado de ejecutar el comando con este modificador es algo parecido a la ventana siguiente:



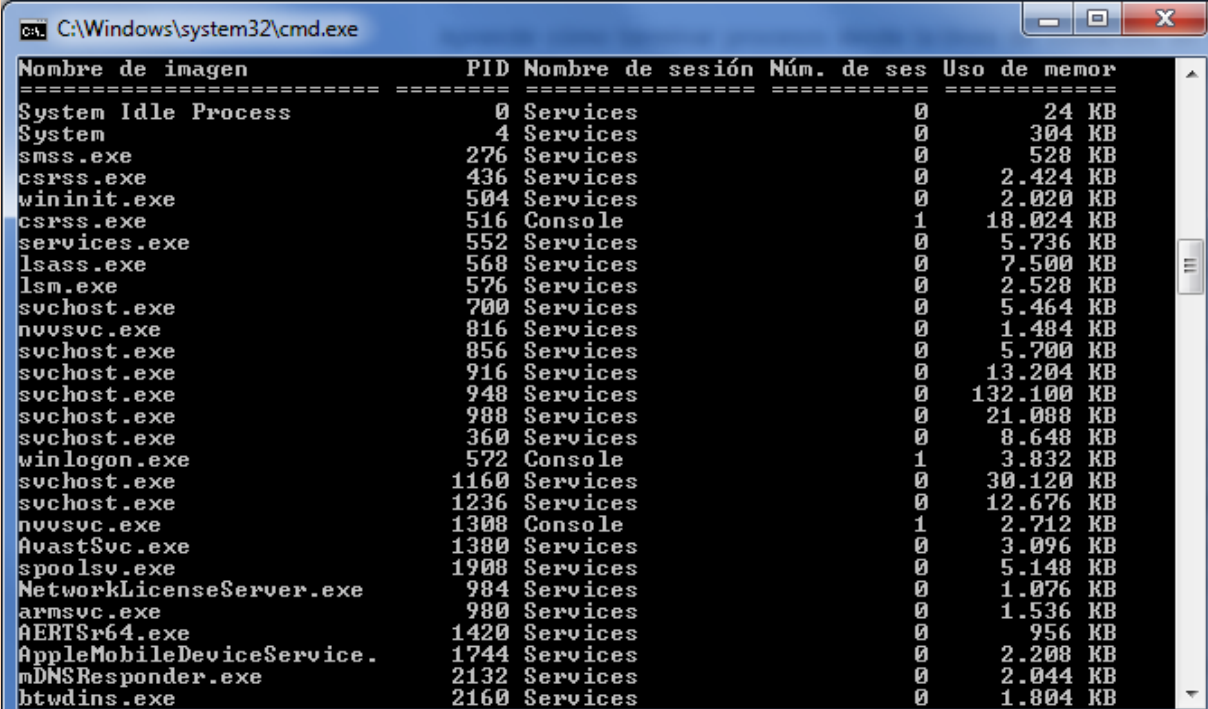
```
alumno1@ubuntu1204-VB: ~  
ps -A  
  PID TTY          TIME CMD  
    1 ?           00:00:05 init  
    2 ?           00:00:00 kthreadd  
    3 ?           00:00:01 ksoftirqd/0  
    5 ?           00:00:03 kworker/u:0  
    6 ?           00:00:00 migration/0  
    7 ?           00:00:00 watchdog/0  
    8 ?           00:00:00 cpuset  
    9 ?           00:00:00 khelper  
   10 ?           00:00:00 kdevtmpfs  
   11 ?           00:00:00 netns  
   12 ?           00:00:00 sync_supers  
   13 ?           00:00:00 bdi-default  
   14 ?           00:00:00 kintegrityd  
   15 ?           00:00:00 kblockd  
   16 ?           00:00:00 ata_sff  
   17 ?           00:00:00 khubd  
   18 ?           00:00:00 md  
   19 ?           00:00:00 kworker/u:1  
   20 ?           00:00:00 khungtaskd  
   21 ?           00:00:01 kswapd0  
   22 ?           00:00:00 ksmd  
   23 ?           00:00:00 khugepaged  
--Más--
```

Si queremos eliminar uno de los procesos, bastaría con **matarlo**, y para ello bastaría con ejecutar el comando **kill** seguido del PID del proceso.

GESTIONAR PROCESOS EN WINDOWS 7

Para conocer los procesos que tenemos ejecutándose en un S.O. basado en Windows 7 tenemos multitud de herramientas en el mercado e incluso algunas ya instaladas en el propio sistema.

Nosotros vamos a elegir la forma genérica que es la que se realiza utilizando la línea de comandos de Windows. Para ello, ejecutamos una ventana de **CLI** y una vez dentro de ella, ejecutamos la orden **tasklist**. Este nos mostrará la siguiente información en la ventana:



Nombre de imagen	PID	Nombre de sesión	Núm. de ses	Uso de memor
System Idle Process	0	Services	0	24 KB
System	4	Services	0	304 KB
smss.exe	276	Services	0	528 KB
csrss.exe	436	Services	0	2.424 KB
wininit.exe	504	Services	0	2.020 KB
csrss.exe	516	Console	1	18.024 KB
services.exe	552	Services	0	5.736 KB
lsass.exe	568	Services	0	7.500 KB
lsm.exe	576	Services	0	2.528 KB
suchost.exe	700	Services	0	5.464 KB
nvsvc.exe	816	Services	0	1.484 KB
suchost.exe	856	Services	0	5.700 KB
suchost.exe	916	Services	0	13.204 KB
suchost.exe	948	Services	0	132.100 KB
suchost.exe	988	Services	0	21.088 KB
suchost.exe	360	Services	0	8.648 KB
winlogon.exe	572	Console	1	3.832 KB
suchost.exe	1160	Services	0	30.120 KB
suchost.exe	1236	Services	0	12.676 KB
nvsvc.exe	1308	Console	1	2.712 KB
AvastSvc.exe	1380	Services	0	3.096 KB
spoolsv.exe	1908	Services	0	5.148 KB
NetworkLicenseServer.exe	984	Services	0	1.076 KB
armsvc.exe	980	Services	0	1.536 KB
AERTSr64.exe	1420	Services	0	956 KB
AppleMobileDeviceService.exe	1744	Services	0	2.208 KB
mDNSResponder.exe	2132	Services	0	2.044 KB
btwdins.exe	2160	Services	0	1.804 KB

Esta forma de ejecutar el comando es la más básica y nos mostrará cinco columnas con el nombre de la imagen(proceso), el PID, el nombre de la sesión, el número de esta sesión y el uso de la memoria que está haciendo el proceso.

Si queremos la información con otra estructura o filtrada, utilizaremos los diferentes modificadores que tiene el comando y puedes estudiar si escribes **tasklist /?**.

Si queremos eliminar uno de los procesos, habría que **matarlo**, y para ello bastaría con ejecutar el comando **taskkill** seguido de **/PID** y el número de PID del proceso que queremos matar.

6.4. Planificación de procesos en la multiprogramación

Los primeros S.O. se denominaban de monotarea o serie; en ellos, hasta que no finalizaba la ejecución de un programa no empieza a ejecutarse otro. Un S.O. multitarea aprovecha los tiempos muertos de la CPU, los tiempos muertos en periféricos y los espacios de memoria principal no ocupados por el proceso. Consiste, en esencia, en cargar en la memoria principal varios procesos e ir asignando la CPU sucesivamente a los distintos procesos en memoria, de forma que se aproveche al máximo la CPU y varios procesos vayan avanzando en su ejecución, sin necesidad de que finalice completamente uno para iniciar la ejecución de otro.

En los sistemas multitarea puros o clásicos, cuando un proceso entra en estado de bloqueado, un módulo del S.O. denominado **distribuidor o dispatcher** pasa el turno de ejecución a uno de los procesos en memoria principal que esté en estado preparado.

Este tipo de multitarea tiene el inconveniente de que un proceso con mucho tiempo de CPU (muchos cálculos) y pocas entradas y salidas puede monopolizar la CPU, hasta que acabe su ejecución. Los sistemas multitarea actuales no necesariamente deben esperar a que un proceso pase al estado de bloqueado para que el distribuidor lo interrumpa y dé el turno a otro proceso que esté preparado.

Un sistema multitarea se denomina de multiprogramación si dispone de técnicas apropiadas de protección de memoria y de control de concurrencia para permitir el acceso compartido a dispositivos de entrada y salida y archivos. Si prevé el uso concurrente de distintos usuarios se denomina multiusuario.

La idea de tiempo compartido es una forma de gestionar la multiprogramación para obtener sistemas multiusuario, que requieren tiempos de respuesta adecuados, dando la ilusión a cada usuario que está trabajando en exclusiva con la máquina. Con rigor, en multiprogramación, se dice que la CPU está ejecutando concurrentemente (no simultáneamente) varios procesos. Para ello el planificador implementa un algoritmo de planificación. Existen varios, aquí explicaremos algunos de ellos:

6.4.1. Algoritmo de rueda, de petición circular, cooperativo o Round Robin :

A cada uno de los procesos en memoria se le asigna un intervalo de tiempo fijo llamado periodo T o quantum, y se cambia de contexto de un proceso a otro, de forma rotatoria, conforme se van consumiendo los quantum. Este algoritmo se utiliza normalmente en la asignación de tiempos para los actuales sistemas operativos multiusuario y multiproceso y sistemas operativos monousuarios y que trabajan en multitarea.

En los S.O. de tiempo compartido, se cambia el contexto además, en el momento en que quede bloqueado.

Ver ejemplos del [Anexo 1](#).

6.4.2. Algoritmo por intervalos de espera

El tiempo de asignación de la CPU para la ejecución de un proceso dependerá de la prioridad que se le asigne a dicho proceso. A mayor prioridad, más tiempo de CPU.

Ver ejemplos del [Anexo 2](#).

6.4.3. Algoritmo FIFO⁴ o FCFS⁵ (Primero en entrar, primero en salir)

Es el más sencillo y también el más ineficaz. Como su nombre indica los procesos toman la CPU y la abandonan al terminar entrando el siguiente que realiza la misma operación. Este es un algoritmo no apropiativo, es decir, cuando un proceso toma la CPU ningún otro proceso puede quitársela hasta que él no finalice. Este algoritmo se utiliza normalmente para la gestión de trabajos de colas en impresión.

Ver ejemplos del [Anexo 3](#).

6.4.4. Algoritmo SJF⁶ (El más corto primero)

En este algoritmo, es el más corto el que se ejecutaría cuando el proceso actual finalizara. Este algoritmo es también no apropiativo.

Ver ejemplos del [Anexo 4](#).

6.4.5. Algoritmo SRNT⁷ (El menor tiempo restante a continuación)

Este algoritmo permite asignar el tiempo de ejecución de forma prioritaria a procesos muy cortos para ejecutarlos en el menor tiempo posible. Si se está ejecutando un proceso más largo, el S.O. le quitará la ejecución de la CPU para asignársela al proceso más corto. De esta forma, el usuario del proceso corto obtendrá resultados en un tiempo mínimo y el del proceso largo casi no notará esta circunstancia.

Ver ejemplos del [Anexo 5](#).

⁴ FIFO: First In First Out

⁵ FCFS: First Come First Served

⁶ SJF: Shortest Job First

⁷ SRNT: Shortest Remaining Time Next

7. Gestión de memoria

La parte del S.O. que administra la memoria es el **administrador de memoria**. Su labor es clara: llevar en un registro las partes de memoria que se están utilizando y las que no. De esta forma, reservará espacio de memoria para los nuevos procesos y liberará el espacio de los procesos que han finalizado.

También se encarga de gestionar el intercambio de datos entre memoria y disco, siempre y cuando los procesos sean tan grandes que no quepan de una sola vez en memoria.

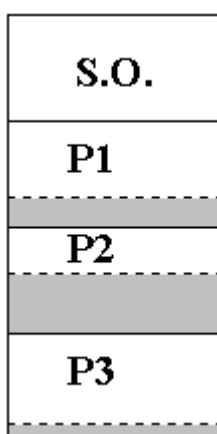
Podemos clasificar los sistemas de administración de memoria en dos grupos:

- Aquellos que desplazan los procesos de memoria central a disco y viceversa.
- Aquellos que no realizan dicho desplazamiento.

7.1. Particiones estáticas

La memoria se divide en cierto número de **particiones o zonas**, cada una de las cuales contendrá un proceso. Las direcciones de base son las direcciones de comienzo de cada partición. El tamaño de la partición es determinado por el operador o por el S.O. (Usualmente 8k, 16k, 32k o 64k). El S.O. mantiene una tabla en la que cada fila corresponde a una partición, conteniendo la posición base de la partición, su tamaño (no todas las particiones tienen que ser iguales) y el estado de la partición. El planificador de trabajos, una vez que una partición está libre, hace que se introduzca el programa de máxima prioridad que hay en la cola de espera y que quepa en dicha partición.

Cuando un trabajo abandona la memoria, su partición queda libre, de forma que será ocupada por otro de menor o igual tamaño. Esto provoca que se fragmente la memoria dejando espacios libres que no pueden ser utilizados por otros programas.



Las zonas sombreadas indican el espacio de cada una de las particiones que queda sin utilizar. Esto es a lo que se denomina **fragmentación de una partición o fragmentación interna**.

Figura 2. Ejemplo de particionamiento estático

7.2. Particiones dinámicas

Los programas, son introducidos por el S.O. inicialmente en posiciones consecutivas de memoria, no existiendo, por tanto, particiones predefinidas.

El S.O. puede gestionar el espacio de memoria usando una **tabla de procesos** en la que cada línea contiene el número de proceso o identificador del mismo, el espacio que ocupa y la dirección base. Existe una tabla complementaria a la anterior con los fragmentos o huecos libres. El planificador de trabajos periódicamente consulta la tabla, introduciendo en memoria los programas que quepan en los fragmentos. Obviamente, al iniciarse una sesión de trabajo se carga el primer programa, dejando un fragmento libre donde se pueden incluir otros programas. Al ir acabando de ejecutarse los procesos, el número de fragmentos crecerá y el espacio de cada uno de ellos disminuirá, llegando un momento en que el porcentaje de memoria aprovechado es muy reducido. El problema puede resolverse haciendo una compactación. Esta consiste en compactar o agrupar todos los fragmentos cuando acaba la ejecución de un proceso, quedando así sólo uno grande. La compactación se efectúa cambiando de sitio o reubicando los procesos en ejecución.

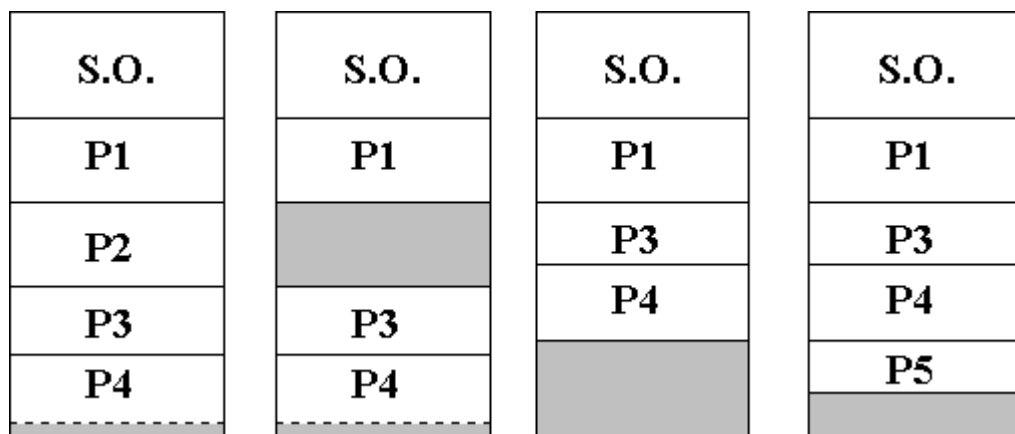


Figura 3. Ejemplo de particionamiento dinámico de memoria

En el ejemplo se ve como en un primer momento todos los procesos se colocan sucesivamente en memoria. Cuando el proceso P2 termina deja un gran espacio (segundo gráfico), una vez realizada la compactación todo el espacio libre queda unido al final de la memoria (tercer gráfico). Por último el nuevo proceso ocupará el espacio que necesite siempre que tenga suficiente.

7.3. Paginación

Con este procedimiento la memoria principal se estructura en **marcos de página** (denominados también **bloques**) de longitud fija. Los procesos de los usuarios se dividen en zonas consecutivas, denominadas **páginas lógicas** o **virtuales** o simplemente **páginas**. Para un sistema dado, la capacidad de página y marco de página son coincidentes, de forma que cada página se memoriza en un marco.

El fundamento de la paginación reside en que no es necesario que un proceso se almacene en posiciones consecutivas de memoria. Las páginas se almacenan en marcos de página libres independientemente de que estén o no contiguos. Cada marco de página contiene instrucciones consecutivas. Una instrucción del proceso se localiza dando el número de marco y la dirección relativa dentro de él.

Ver ejemplo en el [Anexo 6](#).

7.4. Segmentación

En este tipo de gestión de la memoria el funcionamiento es parecido, en cierto modo al de la paginación.

Se denomina **segmento** a un grupo lógico de información, tal como un programa, una subrutina, una pila, una tabla de símbolos, un array o una zona de datos. Esta unidad no es de un tamaño preestablecido, pues depende totalmente del programa de que se trate. Un programa ejecutable es una colección de segmentos.

El programa se considera dividido en sus segmentos por lo que no hay un tamaño fijo de memoria reservado para cada segmento. La gestión la realiza el S.O., como con las particiones dinámicas, sólo que cada partición no corresponde a un proceso sino a un segmento. El S.O. mantiene una tabla-mapa de segmentos similares a la de páginas.

Como el tamaño de los segmentos es variable, antes de realizar un acceso a memoria se comprueba que el desplazamiento no supere al tamaño del segmento, para proteger a las zonas de memoria ocupadas por otro segmento.

La segmentación permite que ciertos procesos puedan compartir código, rutinas, etc. o datos comunes sin necesidad de estar duplicados en memoria principal. Así si seis usuarios están utilizando interactivamente un procesador de textos determinado, no sería necesario que estuviesen cargadas en memoria las seis copias idénticas del código del programa. El S.O. se limitaría a anotar en las tablas-mapas de segmentos de cada uno de los procesos la dirección donde se encuentra el código. Por tanto, en un momento dado, en memoria existirían: segmentos asignados a trabajos concretos, segmentos compartidos "shared" y bloques libres.

Ver ejemplo en [Anexo 7](#).

7.5. Memoria virtual

La técnica de memoria virtual fue diseñada en 1961 por Fotheringham.

La memoria virtual permite a los usuarios hacer programas de una capacidad muy superior a la que físicamente tiene el ordenador. En realidad, la memoria virtual hace posible que la capacidad máxima de los programas venga limitada por el espacio que se reserve en disco para ella y no por la memoria principal. En definitiva, los sistemas con memoria virtual presentan al usuario una memoria principal aparente mayor que la memoria física real.

Por otra parte, permite que aumente el número de procesos en la memoria principal en ejecución concurrente, ya que con ella solo es necesario que esté en memoria principal un trozo mínimo (**página o segmento**) de cada proceso, y no el proceso completo.

Para implantar la memoria virtual suele utilizarse una de las siguientes técnicas de gestión de la memoria física:

- gestión de memoria por páginas,
- gestión de memoria segmentada
- gestión de memoria segmentada-paginada.

La memoria virtual se basa en que las instrucciones de un programa que se ejecutan sucesivamente (en un corto intervalo de tiempo) están en direcciones muy próximas y en que los programas suelen estar redactados con gran linealidad.

En un sistema de memoria virtual se mantiene en disco un archivo con la **imagen del proceso** completo, que está troceado en páginas o segmentos, dependiendo del método. En cambio en la memoria únicamente debe estar la página (o segmento) que en ese momento deba estar en ejecución, intercambiándose páginas entre disco y memoria principal cuando sea necesario.

La gestión de memoria virtual segmentada es más compleja que la del tipo de paginación, ya que los segmentos son de tamaño variables y son más difíciles de gestionar.

La memoria virtual con paginación combina las técnicas de paginación e intercambiabilidad. Por lo general se utiliza un método de intercambiabilidad “perezosa” (“lazy swapper”): únicamente se lleva a memoria una página cuando es necesaria para algún proceso, de esta forma, el número de procesos en ejecución concurrente puede aumentar.

7.6. Swapping

El Swapping es una técnica similar a la de memoria virtual. Cuando varios usuarios están ejecutándose procesos en un mismo ordenador, éste se ve obligado a cargarlos en RAM. Según el estado en el que se encuentre el proceso de cada usuario, la memoria se irá liberando o no. Si un usuario interrumpe por un momento la ejecución de su proceso, pasará a la zona de swap mediante la técnica llamada swap-out. De esta forma, la memoria interna queda liberada para que en ella se pueda almacenar otro proceso del mismo usuario o de otro.

Si el usuario vuelve a solicitar su proceso para seguir ejecutándolo, se produce el denominado swap-in, que consiste en pasar el programa de la zona de swap a la memoria interna.

Esta zona de swap se suele utilizar en S.O. como UNIX y LINUX. Está formada por un espacio físico del disco en el que tenemos el S.O. y las aplicaciones que se van a ejecutar. Los fabricantes de estos S.O. recomiendan que esta zona sea del 20 %, aproximadamente el espacio total en disco o el doble de la capacidad RAM del ordenador.

La diferencia entre la gestión de memoria virtual y el Swapping es que, mediante la primera, puede llegar a ocurrir que el disco esté tan lleno que la gestión sea difícil o imposible, ya que el espacio destinado al intercambio suele ser espacio del disco duro en el que está instalado tanto el S.O. como el software de aplicaciones y los datos del usuario.

En el Swapping no puede ocurrir esto, ya que esta zona siempre estará disponible para el intercambio de programas con la memoria principal. Normalmente, al estar esta zona en un dispositivo físico diferente, todo el espacio estará disponible cada vez que encendamos el ordenador.

8. Gestión de archivos

Un archivo, dependiendo principalmente del uso o capacidad que vaya a tener, puede estructurarse en un soporte de distintas formas. Hay un sistema físico más o menos complejo, y una visión lógica adecuada al usuario. El S.O. hace posible utilizar esta última, haciendo de interfaz entre las dos. En efecto, desde el punto de vista hardware, para almacenar datos o programas sólo existen direcciones físicas. En un disco toda información se graba o lee en bloques referenciados por (nº de unidad)/(superficie)/(pista)/(sector). El S.O. posibilita que el usuario no tenga que utilizar direcciones físicas: podemos actuar sobre un archivo y almacenar o recuperar información sin más que indicar su nombre y utilizar ciertas instrucciones del lenguaje de control del S.O.

Sobre la estructura hardware descrita, el S.O. construye dos abstracciones: la de archivo y la de directorio (carpetas en el S.O. Windows). El conjunto de módulos del S.O. que se encarga de la gestión de archivos y directorios se suele denominar **sistema de archivos**.

El concepto de archivo posibilita aislar al usuario de los problemas físicos de almacenamiento. Cuando el usuario desee referirse a un conjunto de información del mismo tipo como una unidad de almacenamiento, no tiene nada más que crear un archivo dándole el nombre que considere oportuno. Los archivos se conciben como estructuras con las siguientes peculiaridades:

- Deben ser capaces de contener grandes cantidades de información.
- Su información debe permanecer y sobrevivir a los procesos que la generan o utilizan.
- Distintos procesos deben poder acceder a la información del archivo concurrentemente.

Cada archivo usualmente contiene su nombre, atributos y los datos.

Los atributos pueden incluir cuestiones tales como fecha y hora de creación, fecha y hora de la última actualización, bits de protección (solo lectura o lectura y escritura), contraseña de acceso, número de bytes por registro, capacidad máxima del archivo y capacidad actualmente ocupada.

Los datos se almacenan en el dispositivo de memoria masiva en forma de bloques. El dispositivo más usado para almacenamiento de archivos es el disco. Como la unidad física de almacenamiento es el bloque, estos pueden grabarse de diversas formas:

- **Contigua:** Los bloques se almacenan uno detrás de otro.
- **Lista encadenada:** los bloques se almacenan en cluster no necesariamente consecutivos, encadenándose un bloque con otro por medio de un puntero. Se accede al archivo conociendo la dirección del “cluster” inicial.
- **Lista de enlaces:** Cada disco dispone de una tabla con tantos elementos como bloques físicos, la posición de cada elemento se corresponde biunívocamente con cada bloque y contiene el puntero del lugar donde se encuentra el siguiente bloque de archivo. Cuando se abre un archivo, el sistema de archivos carga en la memoria principal la lista de enlaces, pudiéndose obtener rápidamente las direcciones de los bloques

consecutivos del archivo. Presenta el inconveniente de que si el disco es muy grande, la lista de enlaces ocupa una capacidad excesiva en memoria principal. Es el sistema utilizado por MSDOS y los sistemas operativos Windows, que usan tablas FAT, tablas FAT32, y tablas NTFS.

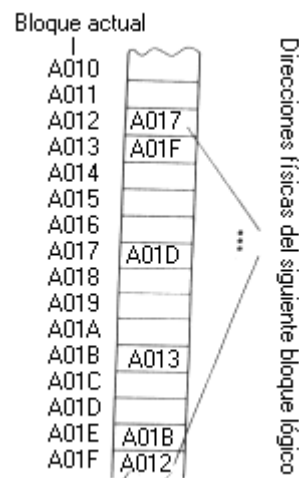


Figura 7. Listas de enlaces

- **I-nodos:** Corresponde a la forma de gestionar los archivos por el S.O. UNIX. Cada archivo tiene asociado un nodo de índices o i-nodos que es una pequeña tabla de tamaño fijo conteniendo los atributos del archivo y las direcciones de un número determinado de los primeros bloques de archivo. Los tres últimos elementos de la tabla indican las siguientes direcciones de los bloques de archivo, pero de forma indirecta: con simple indirección, indirección doble y triple indirección.

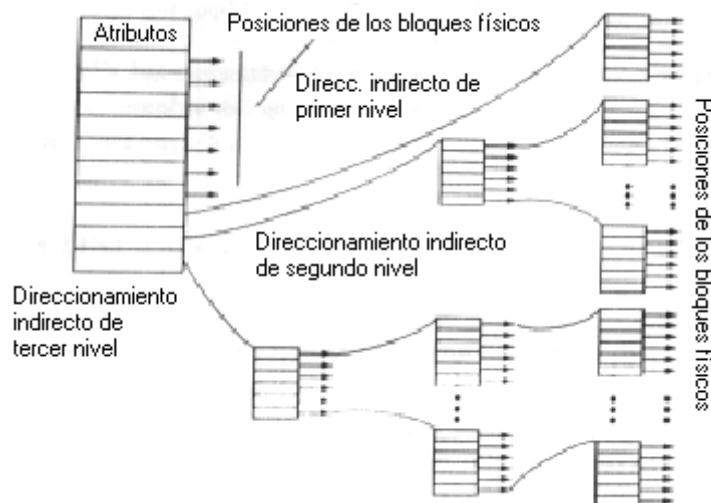


Figura 8. Estructura de i-nodos

La segunda abstracción que utiliza el S.O. para gestionar volúmenes de datos es la de directorio. Los directorios son conjuntos de archivos agrupados, siguiendo algún criterio arbitrariamente elegido: directorio del usuario que lo crea, directorio de facturas, etc.

Un directorio se gestiona con una tabla índice que contiene un elemento por cada archivo o directorio dependiente de él. Cada elemento está formado por el nombre del

archivo dado por el usuario e información adicional. La información adicional sobre el archivo puede estar constituida por los atributos del archivo y el bloque donde comienza el archivo, caso de MSDOS. En el caso de UNIX, en el que el puntero sencillamente es la dirección del i-nodo del archivo, que, como hemos visto anteriormente, contiene tanto los atributos del archivo como la tabla de posiciones.

8.1. Sistemas de archivos FAT

Lo que actualmente conocemos por FAT es realmente FAT16. Es el sistema de archivos introducido por Microsoft en 1.987 para dar soporte a los archivos de 16bits, no soportados por versiones anteriores de FAT (FAT12).

Este sistema de archivos tiene una serie muy importante de limitaciones, entre las que destacan:

- Tamaño máximo de particiones que se pueden gestionar de 2 Gb
- Tamaño máximo de archivos que puede gestionar de 4 Gb.
- Utilización de cluster de 32 o 64 Kb.

Nombres de archivos limitados al formato 8+3 (8 dígitos para el nombre + 3 de extensión). El nombre debe empezar con una letra o número y puede contener cualquier carácter ASCII excepto .(punto), "(comillas), /, \, [,], : (dos puntos), ;(punto y coma), |(Barra), =(igual) o ,(coma)

➤

En 1.996, junto con la salida al mercado del Windows 95 OSR2, se introduce el sistema de archivos FAT32, para solucionar en buena parte las deficiencias que presentaba FAT16, pero manteniendo la compatibilidad en modo real con MS-DOS.

Entre estas se encuentra la de superar el límite de 2Gb en las particiones, si bien se mantiene el tamaño máximo de archivo, que es de 4Gb.

Para solucionar este problema, FAT32 utiliza un direccionamiento de cluster de 32bits, lo que en teoría podría permitir manejar particiones cercanas a los 2 Tib (Terabytes), pero en la práctica Microsoft limitó estas en un primer momento a unos 124Gb, fijando posteriormente el tamaño máximo de una partición en FAT32 en 32Gb.

Esto se debe más que nada a una serie de limitaciones del Scandisk de Microsoft, ya que FAT32 puede manejar particiones mayores creadas con programas de otros fabricantes. Un claro ejemplo de esto lo tenemos en los discos externos multimedia, que están formateados en FAT32 a pesar de ser particiones de bastante tamaño (en muchos casos más de 300Gb).

El tamaño del cluster utilizado sigue siendo de 32Kb, lo que sigue significando un importante desperdicio de disco, ya que un archivo de 1Kb está ocupando en realidad 32Kb de disco.

El paso de FAT16 a FAT32 se tenía que realizar en un principio formateando el disco, situación que se mantuvo hasta la salida de Windows 98, que incorporaba una herramienta para pasar de FAT16 a FAT32 sin necesidad de formatear el disco.

Estos dos formatos, a pesar de sus inconvenientes, tienen una gran ventaja, y es que son accesibles (cuando menos para lectura) por una gran cantidad de sistemas

operativos, entre los que destacan Unix, Linux, Mac OS, etc. Esta compatibilidad es aun mayor en FAT16 que en FAT32.

El sistema de archivos FAT se caracteriza por la tabla de asignación de archivos (FAT), que en realidad es una tabla en la que reside la parte "superior" de la partición. Para proteger la partición, se conservan dos copias de la FAT por si una de ellas resulta dañada. Además, las tablas de FAT y el directorio raíz deben almacenarse en una ubicación fija para que se puedan encontrar correctamente los archivos de inicio del sistema.

8.2. Sistemas de archivos NTFS

El sistema de archivos NTFS, o New Technology File System fue introducido a mediados de 1.993 en Windows NT 3.1, y utilizado por Microsoft solo en sus sistemas profesionales hasta la salida de Windows XP, que fue el primer sistema operativo de uso doméstico que lo incorporó.

Este sistema de archivos tiene una gran serie de ventajas, incluida la de soportar compresión nativa de ficheros y cifrado (a partir de Windows 2000).

También permite por fin gestionar archivos de más de 4Gb, fijándose el tamaño máximo de estos en unos 16Tb. En cuanto a las particiones, permite un tamaño de hasta 256Tb.

Utiliza cluster de 4Kb (aunque se pueden definir de hasta 512bytes, es decir, 1 sector por cluster). Esto permite un aprovechamiento del disco mucho mayor que en FAT16 o en FAT32, ya que, siguiendo el ejemplo anterior de un fichero de 1Kb, si el tamaño del cluster es de 4Kb estaríamos desperdiciando solo 3Kb, y si el tamaño del cluster fuera de 512bytes, pues utilizaría dos cluster, no existiendo en ese caso ningún desperdicio de espacio (hay que considerar que el FAT32 se desperdiciarían 31Kb por cada archivo de 1Kb que tengamos).

Pero tiene un inconveniente, y es el de que en ese caso se necesita un espacio del disco bastante grande para guardar la información del formato. Hay que pensar que con este sistema, a igualdad de espacio (32Kb), para una partición NTFS basada en cluster de 4Kb tendremos ocho cluster en vez de uno solo. Esto en la practica quiere decir que para un archivo de 32Kb hay que guardar 8 direcciones en vez de una sola, pero un simple vistazo a nuestro disco duro nos permite darnos cuenta de que, a pesar de esta pérdida inicial de espacio, en la práctica tenemos una muy superior capacidad de almacenamiento, ya que el espacio desperdiciado es muchísimo menos.

Las particiones formateadas en NTFS no son accesibles desde MS-DOS, Windows 95, Windows 98 ni por otros sistemas operativos instalados en discos bajo sistemas FAT16 o FAT32. Linux tiene soporte parcial de escritura y total de lectura para particiones NTFS.

En realidad, lo que muchos llaman MS-DOS en Windows XP es tan solo un editor de comandos, con un emulador de MS-DOS para poder ejecutar algunos programas basados en DOS (no todos), eso si, de 16bits, ya que NTFS no tiene soporte para programas de 8bits.

Se puede pasar muy fácilmente una partición FAT32 a NTFS sin pérdida de datos, mediante comandos de consola.

Podemos pasar mediante software de FAT16 a FAT32 y de este a NTFS sin pérdida de información ni de nada (teniendo en cuenta siempre los riesgos que un cambio de formato de partición implican), pero no a la inversa.

8.3. Sistemas de archivos ext para Linux

El sistema ext2 (Second extended filesystem, Segundo sistema de archivos extendidos) admite particiones en el disco de hasta 4 TB y archivos de hasta 2 GB, se podría utilizar nombres de hasta 255 caracteres y es muy estable.

El ext3 (Tercer sistema de archivos extendidos), no es una versión nueva sino una modificación de ext2 que como características principales indicaremos que admitía *journaling* (esto es, hacer un registro de los cambios realizados en el sistema de ficheros).

Desde Diciembre de 2008 a partir de la versión del kernel 2.6.28 está disponible ext4 (Cuarto sistema de archivos extendidos). Además ser un sistema de archivos transaccional (admite journaling), ofrece también las siguientes ventajas:

- Soporte de volúmenes de hasta 1024 PB⁸
- Soporta extents.
- Menor uso de la CPU
- Mejoras en la velocidad de lectura y escritura

Los sistemas ext tiene limitación de 256 caracteres alfanuméricos para el nombre aunque no usan la extensiones para identificar el tipo de archivo. Además son sistemas que distinguen entre mayúsculas y minúsculas.

Tanto ext2 como ext3 pueden gestionar volúmenes de hasta 4TB, teniendo en cuenta de que esto también depende del tamaño del blocksize elegido. Además los archivos pueden tener un tamaño máximo de 2 TB. Con ext4 podemos alcanzar los 1024 PB con archivos de hasta 16 TB.

8.4. Sistemas de archivos HFS para MAC

El sistema de archivos Mac OS Extended (HFS Plus) reemplaza al HFS (Hierarquical File System, Sistema de archivos jerárquicos). También se denomina Mac OS Standard. Los ficheros de este sistema usan Unicode y pueden llegar hasta 255 caracteres.

8.5. Otros sistemas de archivos

- **HPFS (High Performance File System):** fue creado específicamente para el sistema operativo OS/2 para mejorar las limitaciones del sistema de archivos FAT. Se caracterizaba por permitir nombres largos, metadatos e información de seguridad, así como de autocomprobación e información estructural. Otra de sus características es que, aunque poseía tabla de

⁸ PB: Petabyte

archivos (como FAT), ésta se encontraba posicionada físicamente en el centro de la partición, de tal manera que redundaba en menores tiempos de acceso a la hora de leerla/escribirla.

- **CDFS (CD-ROM File System):** Sistema de archivos desarrollado únicamente para montarse sobre CDROM's.
- **Unix File System (UFS):** es un sistema de archivos utilizado por varios sistemas operativos UNIX y POSIX. Es un derivado del Berkeley Fast File System (**FFS**), el cual es desarrollado desde *FS* UNIX (este último desarrollado en los Laboratorios Bell). Casi todos los derivados de BSD incluyendo a FreeBSD, NetBSD, OpenBSD, NeXTStep, y Solaris utilizan una variante de UFS. En Mac OS X está disponible como una alternativa al HFS. En Linux, existe soporte parcial al sistema de archivos UFS, de solo lectura, y utiliza sistema de archivos nativo de tipo ext3, con un diseño inspirado en UFS.
- **Universal Disk Format (UDF)** es un sistema de archivos con estándar ISO 9660 propiedad de Adaptec que utiliza las grabadoras de CD/DVD como un dispositivo de almacenamiento lógico. Este formato permite leer, escribir o modificar los archivos contenidos en discos CD/DVD reescribibles (RW).

9. Gestión de entrada y salida.

Una de las funciones principales de un S.O. es el control de los periféricos de entrada/salida del ordenador. El S.O. se encarga de enviar órdenes, determinar el dispositivo que necesita la atención del procesador, eliminar posibles errores, etc.

En primer lugar, tenemos que hacer una clasificación de los periféricos. Esta clasificación no corresponde a si son periféricos de entrada o de salida; hay que clasificarlos según gestionen la información por bloques o por caracteres. La clasificación es la siguiente:

- Periféricos tipo bloque: Son aquellos en los que la información que se maneja es de tamaño fijo. La información entra o sale de memoria en forma de bloque. Un ejemplo son los registros de ficheros de datos almacenados en discos o disquetes, ya que cada registro contiene información referente a un bloque homogéneo.
- Periféricos tipo carácter: Son los que sirven para introducir datos en forma de caracteres, sin ningún orden concreto, dentro de la memoria del ordenador como pueden ser los teclados, el ratón, etc...

Cada periférico está compuesto por un componente mecánico y por otro, u otros, componentes electrónicos y por la denominada controladora o adaptador, encargado de conectar el dispositivo físico al ordenador.

El S.O. se encarga de acceder a la información de la memoria principal, extraerla en forma de impulsos eléctricos y enviarla a los diferentes dispositivos periféricos. Si la información se envía a un HHDD, los impulsos se transformarán en señales de tipo magnético; si son enviados a la impresora, se transformará en caracteres impresos, etc...

Los dispositivos físicos que el S.O. tiene que gestionar para que la información pase de un sitio al otro del ordenador se clasifican según la función que realizan en:

- Soportes de almacenamiento
- Interfaces o periféricos propiamente dichos.
- Soportes de transmisión: buses y canales.

Hay que destacar las interfaces como medio de comunicación entre hardware y software a través del S.O. Que pueden ser :

- Interfaz tipo texto.
- Interfaz tipo gráfico.

Para la gestión de periféricos hay que hablar de controladora, de canal y de interrupción. La controladora es un componente de hardware que sirve para gestionar el uso de periféricos. Las controladoras o adaptadores, vistos en la unidad de trabajo 1, son de varios tipos; su función, como ya vimos, es conectar físicamente el periféricos a la placa base del ordenador para que exista comunicación.

Las controladoras necesitan un pequeño software para que exista comunicación entre el periférico y el microprocesador. Este software, llamado **driver** (o controlador), se encarga de realizar funciones de traducción entre el periférico y el ordenador para que ambos “se entiendan”. Los drivers suelen suministrarlos los fabricantes de periféricos y vienen diseñados para varios S.O.

Los buses, líneas o canales (vistos también en la unidad de trabajo 1), son los circuitos o “caminos” por los cuales se envía y recibe la información entre los diferentes elementos del ordenador.

9.1. Interrupciones

A las interrupciones se las conoce como IRQ (Interrupt **R**equest) y tienen como función lo que su nombre indica: interrumpir el trabajo del procesador para destinarlo a otra actividad. Estas interrupciones solamente se envían en caso necesario a través de las líneas de señales, que en realidad son las propias IRQ.

Cada una de ellas es provocada por el periférico que la necesita. Cuando pulsamos una tecla estamos provocando una interrupción. Entonces, la CPU deja de hacer otras tareas y dedica su atención al elemento que lanzó la interrupción, en este caso al teclado, mostrando el carácter pulsado en el monitor.

Un ordenador habitualmente cuenta con 16 posibles interrupciones, numeradas del 0 al 15. Gracias a este número, el procesador sabe qué periférico ha provocado la interrupción. Entonces deja lo que está haciendo y atiende la solicitud realizada por el dispositivo. Algunas de estas 16 están reservada al propio sistema.

Siempre que instalemos alguna nueva tarjeta de expansión, es necesario que le asignemos una interrupción. La asignación de las interrupciones a los diferentes dispositivos puede hacerse mediante hardware, mediante **jumpers**, en este caso no podrá ser modificada a no ser que se modifique la posición de este jumpers o software, que actualmente es lo más habitual.

Las IRQ no pueden ser usadas por más de un dispositivo.

IRQ	Ancho del Bus	Función asignada
0	8	Reloj del sistema
1	8	Teclado
2	8	Controlador de interrupciones
3	8	COM2
4	8	COM1
5	8	LPT2
6	8	Unidad de disquete
7	8	LPT1
8	8	Reloj de tiempo real
9	16	Remite al IRQ2
10	16	Disponible
11	16	Disponible
12	16	Disponible
13	16	Coprocesador
14	16	HHDD
15	16	Disponible

Figura 9. Tabla de interrupciones

En la tabla anterior vemos la asignación habitual de un PC. Como vemos solo existen en principio cuatro IRQ disponibles la 10,11,12 y 15. ¿Qué ocurre si necesitamos instalar 5 dispositivos aun no asignados en nuestro equipo? La solución pasa por mirar cuales de los dispositivos asignados no está siendo usado.

Actualmente, muchos ordenadores no usan los puertos COM1 y COM2, incluso con los puertos USB el puerto LPT1 y el LPT2 suelen estar libres. En este caso, podríamos asignar estas interrupciones a otros dispositivos. Esto provocaría un conflicto de interrupciones en el momento en que conectemos alguno de estos dispositivos. Si todas las interrupciones están ocupadas, solo podremos instalar dispositivos nuevos en las cuatro interrupciones libres.

9.2. DMA⁹ o Acceso Directo a Memoria

El acceso directo a memoria (**Direct Memory Access**) permite a cierto tipo de componentes de ordenador acceder a la memoria del sistema para leer o escribir independientemente de la CPU principal. Muchos sistemas hardware utilizan DMA, incluyendo controladores de unidades de disco, tarjetas gráficas y tarjetas de sonido. DMA es una característica esencial en todos los ordenadores modernos, ya que permite a dispositivos de diferentes velocidades comunicarse sin someter a la CPU a una carga masiva de interrupciones.

Una transferencia DMA consiste principalmente en copiar un bloque de memoria de un dispositivo a otro. En lugar de que la CPU inicie la transferencia, la transferencia se lleva a cabo por el controlador DMA. Un ejemplo típico es mover un bloque de memoria desde una memoria externa a una interna más rápida. Tal operación no ocupa al procesador y como resultado éste puede ser planificado para efectuar otras tareas. Las transferencias DMA son esenciales para aumentar el rendimiento de aplicaciones que requieran muchos recursos.

Cabe destacar que aunque no se necesite a la CPU para la transacción de datos, sí que se necesita el bus del sistema (tanto bus de datos como bus de direcciones), por lo que existen diferentes estrategias para regular su uso, permitiendo así que no quede totalmente acaparado por el controlador DMA.

9.3. Gestión de dispositivos de entrada y salida lentos

La velocidad del procesador es muy superior a la de los dispositivos de entrada y salida. Algunas soluciones que se han utilizado para contrarrestar la lentitud de los dispositivos han sido utilizar técnicas de almacenamiento intermedio como las siguientes:

- **Caching:** Consiste en guardar una copia de los datos que se utilizan con mayor frecuencia.
- **Buffering:** Es una técnica que se utiliza para contrarrestar la lentitud de los dispositivos E/S en comparación con la CPU. Consiste en mantener un buffer o registro intermedio para lectura y otro para escritura. La CPU y el dispositivo leen o escriben sobre este buffer y lo atienden cuando haya datos en ellos. El principal problema que presenta esta técnica es que no detecta cuándo ha terminado un dispositivo una operación. Para ello se utiliza junto con las interrupciones que le permiten indicar cuando ha finalizado una operación.
- **Spooling:** El Spooling (Simultaneous Peripheral Operation On Line) consiste en utilizar un buffer de gran tamaño (generalmente un disco) para tratar de leer la mayor cantidad posible de información de los dispositivos de entrada

⁹ http://es.wikipedia.org/wiki/Acceso_directo_a_memoria

y almacenar la mayor cantidad posible de información hasta que los dispositivos de salida estén disponibles.

10. Bibliografía y Webgrafía

- **Sistemas informáticos multiusuarios y en red (2002).** Ed. Paraninfo. Saturnino Peña, Mariano Moreno y Carlos Elvira.
- **Sistemas operativos monopuesto (2009).** Ed. McGraw-Hill. Francisco Javier Muñoz.
- **Sistemas operativos monopuesto (2011).** Ed. Editex. Jesús Niño Camazón.
- **Introducción a la Informática (2001).** Ed. Anaya. Jorge Rodríguez Vega
- **Administración de Sistemas GNU/Linux (2010).** Ed. StarBook. Julio Gómez López
- **Microsoft Windows 2000 Profesional.(2001)** Instalación y configuración. Ed. Prentice Hall. Jim Boyce.
- Microsoft Suport <http://support.microsoft.com/kb/100108/es>
- Diferencias entre FAT, FAT32 y NTFS.
<http://www.configurarequipo.com/doc563.html>

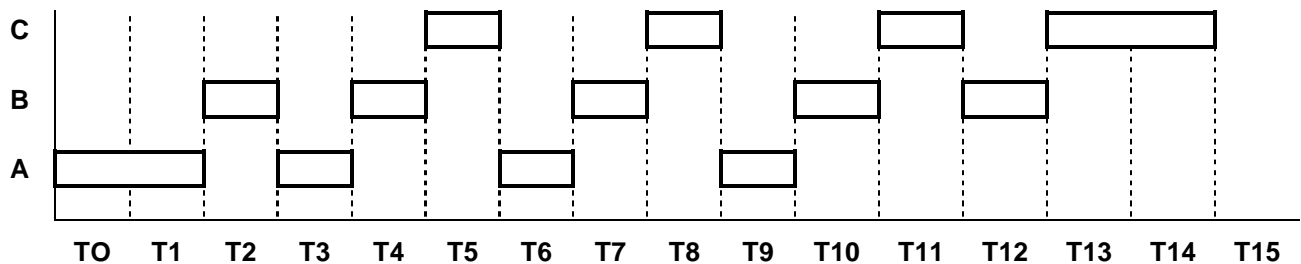
11. ANEXO 1. Algoritmo Round Robin

11.1. Round Robin sin E/S

Supongamos que hay 3 procesos A,B,y C que van a entrar a ejecutarse.

- El proceso A entra en el instante T0 y consumirá 5 quantum;
- El proceso B que consumirá 5 quantum también entra en el instante T2.
- El proceso C entra en el instante T5 y consume 5 quantum de CPU.

El diagrama de cómo se ejecutarían los procesos en la CPU sería el siguiente.

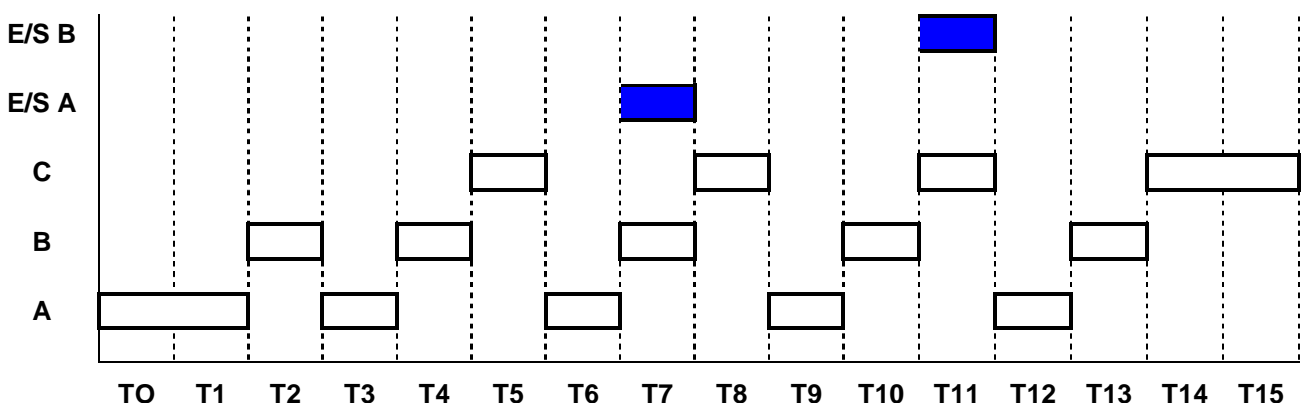


11.2. Round Robin con E/S

Supongamos que hay 3 procesos A,B,y C que van a entrar a ejecutarse.

- El proceso A entra en el instante T0 y consumirá 5 quantum;
- El proceso B que consumirá 5 quantum también entra en el instante T2.
- El proceso C entra en el instante T5 y consume 5 quantum de CPU.
- El proceso A tiene una E/S de datos tras su cuarto quantum de ejecución que dura 1 quantum.
- El proceso B tiene una E/S de datos que dura un quantum tras su cuarto quantum de ejecución.

El diagrama de cómo se ejecutarían los procesos en la CPU sería el siguiente.

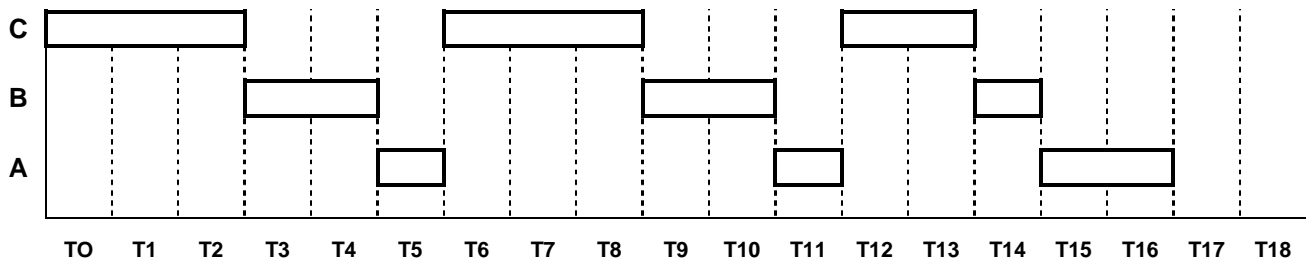


12. ANEXO 2. Algoritmo por intervalos de espera

Supongamos que hay 3 procesos A,B,y C que van a entrar a ejecutarse.

- El proceso C entra en el instante T0 y consumirá 8 quantum;
- El proceso B que consumirá 5 quantum también, entra en el instante T3.
- El proceso A entra en el instante T4 y consume 4 quantum de CPU.
- El proceso C tiene la prioridad máxima mientras que el B tiene una prioridad media y el proceso A la menor prioridad.

El diagrama de cómo se ejecutarían los procesos en la CPU sería el siguiente.

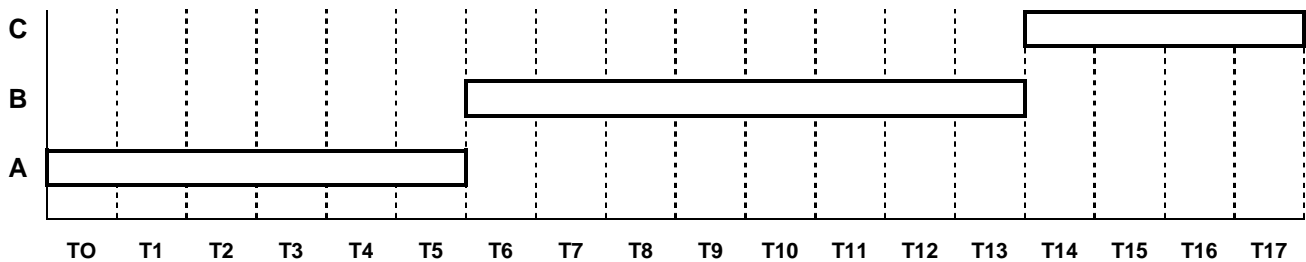


13. ANEXO 3. Algoritmo FIFO ó FCFS

Supongamos que hay 3 procesos A,B,y C que van a entrar a ejecutarse.

- El proceso A entra en el instante T0 y consumirá 6 quantum;
- El proceso B que consumirá 8 quantum también, entra en el instante T3.
- El proceso C entra en el instante T4 y consume 4 quantum de CPU.

El diagrama de cómo se ejecutarían los procesos en la CPU sería el siguiente.



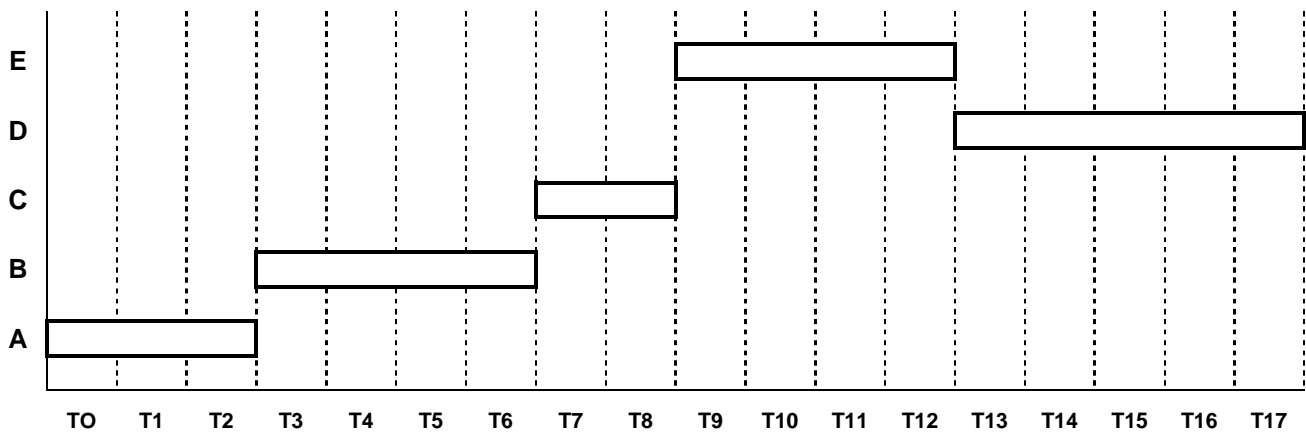
Como puede verse, una vez que un proceso entra en ejecución hasta que no termina su ejecución completa no permite a otro proceso que tome el control de la CPU. Si los procesos, tal y como se ve en el ejemplo, que entran primero son los más largos acapararán todo el tiempo y los procesos cortos tardarán un tiempo excesivo en ejecutarse.

14. ANEXO 4. Algoritmo SFJ

Supongamos que hay 5 procesos A,B,C,D y E que van a entrar a ejecutarse.

- El proceso A entra en el instante T0 y consumirá 3 quantum;
- El proceso B entra en el instante T1 y consumirá 4 quantum
- El proceso C entra en el instante T4 y consumirá 2 quantum
- El proceso D entra en el instante T5 y consumirá 5 quantum
- El proceso E entra en el instante T8 y consumirá 4 quantum

El diagrama de cómo se ejecutarían los procesos en la CPU sería el siguiente.

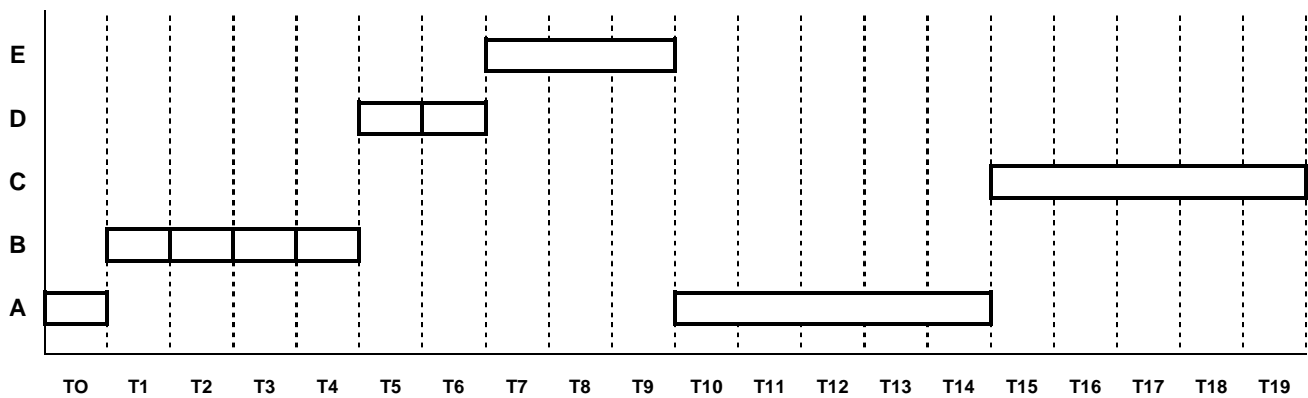


15. ANEXO 5. Algoritmo SRNT

Supongamos que hay 3 procesos A,B,C,D y E que van a entrar a ejecutarse.

- El proceso A entra en el instante T0 y consumirá 6 quantum;
- El proceso B entra en el instante T1 y consumirá 4 quantum
- El proceso C entra en el instante T2 y consumirá 5 quantum
- El proceso D entra en el instante T3 y consumirá 2 quantum
- El proceso E entra en el instante T4 y consumirá 3 quantum

El diagrama de cómo se ejecutarían los procesos en la CPU sería el siguiente.



16. ANEXO 6. Ejemplo de paginación

TABLA-MAPA DE MEMORIA			TABLA DE PROCESOS		
Nº de marco	PID	Dirección base marco	Proceso nº	Tamaño	Ubicación mapa-página
0	S.O.	00000	1	95k	80000
1	S.O.	08000	2	50k	84000
2	S.O.	10000	3	90k	88000
3	P1	18000	4	15k	8C000
4	P1	20000	5	66k	90000
5	P3	28000			
6	P3	30000			
7	P2	38000			
8	P5	40000			
9	libre	48000			
10	P4	50000			
11	P5	58000			
12	P5	60000			
13	P1	68000			
14	P2	70000			
15	P3	78000			

Proceso 1		Proceso 2		Proceso 3	
Página	Marco	Página	Marco	Página	Marco
0	3	0	7	0	15
1	4	1	14	1	5
2	13			2	6

Proceso 4		Proceso 5	
Dirección	Marco	Página	Marco
0	10	0	8
		1	11
		2	12

Figura 4. Ejemplo de tablas de paginación

17. ANEXO 7. Ejemplo de Segmentación

Supongamos un ordenador que admite hasta 256 segmentos de cómo máximo 4Kb cada uno. Las direcciones lógicas estarán compuestas de 20 bits, los 8 primeros identificarán el segmento y los 12 restantes identificarán el desplazamiento dentro del segmento. La dirección lógica A37FF se transforma en una dirección física según el esquema que se muestra en la figura siguiente.

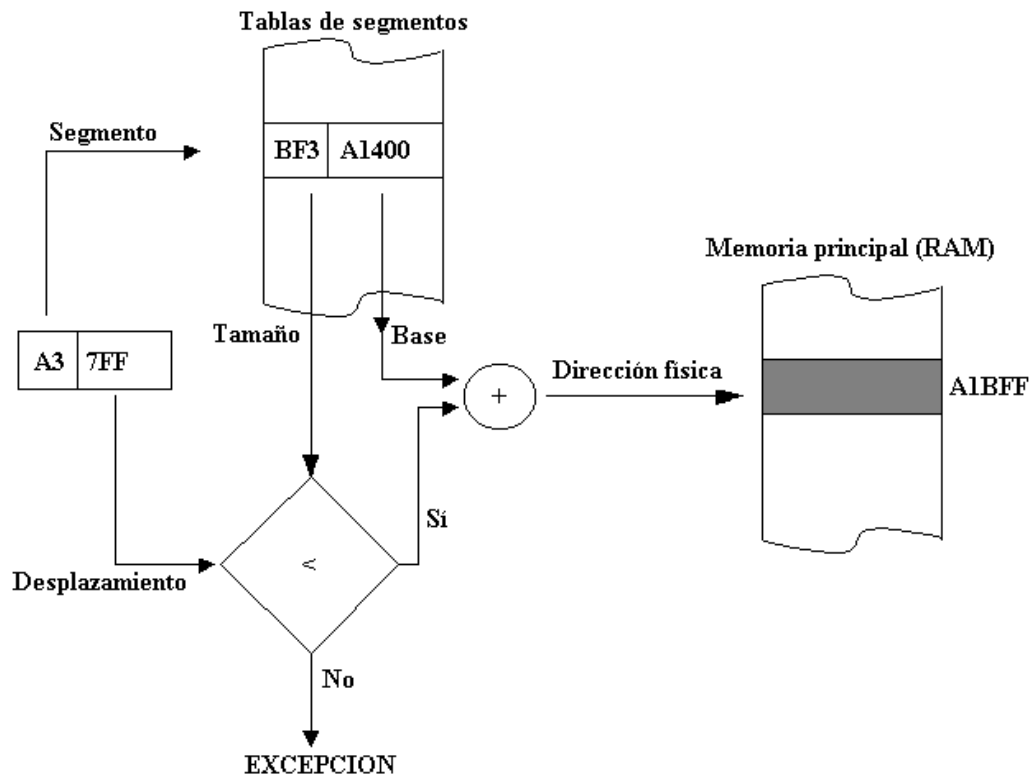


Figura 5. Esquema de búsqueda de direcciones físicas con segmentación.

En el ejemplo, A3 es la dirección de la entrada en la tabla de segmentos. 7FF es la dirección virtual que debe ser menor que BF3 que corresponde al tamaño del segmento. Si fuese mayor 7FF que BF3 se produciría una excepción o interrupción del proceso. En este caso si es menor la dirección virtual que el tamaño del segmento por lo cual, esta dirección virtual se sumará a la dirección base (A1400) para encontrar la dirección física donde se almacenará el segmento en memoria ($A1BFF = A1400 + 7FF$).